(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 August 2001 (23.08.2001)

PCT

(10) International Publication Number
WO 01/61547 A2

(51) International Patent Classification[7]: G06F 17/00

(21) International Application Number: PCT/US01/05337

(22) International Filing Date: 20 February 2001 (20.02.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/507,710 18 February 2000 (18.02.2000) US

(71) Applicant: MARKET ENGINE CORPORATION [US/US]; Suite 410, 2855 Telegraph Avenue, Berkeley, CA 94705 (US).

(72) Inventors: BLASER, Rico; 1309 F Gate View Avenue, San Francisco, CA 94130 (US). MOUTCHKINE, Andrei; 2620 Hillegass Avenue #10, Berkeley, CA 94704 (US). ATAEE, Behrooz; 30302 Meridien Circle, Union City, CA 94587 (US).

(74) Agent: LOVEJOY, David, E.; Fliesler Dubb Meyer & Lovejoy LLP, Four Embarcadero Center, Fourth Floor, San Francisco, CA 94111-4156 (US).

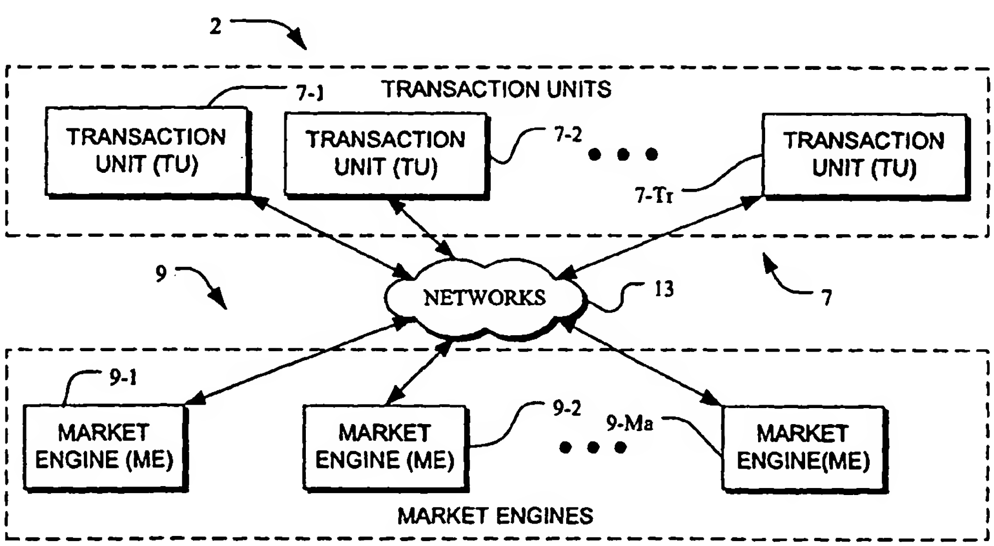
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: MARKET ENGINE HAVING CUSTOMIZABLE CROSSING COMPONENTS



(57) Abstract: A crossing method and apparatus for an e-commerce system for electronic transactions. Electronic transactions involve participants having interests in instruments to be crossed. A crossing unit stores variables for use in mapping the interests of the participants, stores constraints for limiting the mapping of interests of the participants and stores an expression relating the variables. The crossing unit operates to determine values of the variables that optimize the expression, while satisfying the constraints, thereby mapping the interests of the participants.

WO 01/61547 A2

TITLE

MARKET ENGINE HAVING CUSTOMIZABLE CROSSING COMPONENTS

CROSS-REFERENCES

This application is a continuation-in-part of the application entitled ELECTRONIC SYSTEMS WITH SUPERVISION OF TRANSACTIONS AMONG TRANSACTION INITIATORS AND TRANSACTION PROCESSORS, invented by Rico Blaser and Andrei Moutchkine, filed July 26, 1999 and having SC/Serial No. 09/360,899.

This application is a continuation-in-part of the application entitled ELECTRONIC SYSTEMS FORMED OF MARKET ENGINES HAVING INTEGRATED TRANSACTION UNITS, invented by Rico (NMI) Blaser; Andrei (NMI) Moutchkine; Chad Owen Yoshikawa; and Behrooz (NMI) Ataee; SC/Serial No. 09/391,583; Filing Date: September 8, 1999.

This application is a continuation-in-part of the application entitiled MARKET ENGINES HAVING EXTENDABLE COMPONENT ARCHITECTURE, invented by Rico (NMI) Blaser, filed January 26, 2000 and having SC/Serial No. 09/491,074.

COPYRIGHT NOTICE

BACKGROUND OF THE INVENTION

The present invention relates to the field of electronic commerce (e-commerce) and particularly to electronic systems for supervising transactions among multiple transaction initiators and multiple transaction processors and for internal and external crossing of orders in capital and other e-commerce markets.

Alternative trading systems (ATS) have an increasing presence in the securities markets. Examples of alternative trading systems include Instinet, Optimark, Attain, Archipelago, Island, REDI, Posit and the Arizona Stock

Optimark, Attain, Archipelago, Island, REDI, Posit and the Arizona Stock Exchange. Dramatic growth in the number of alternative trading systems and in the volume of securities traded by alternative trading systems have permitted companies to offer faster, less expensive and more flexible ways for investors to trade. This growth has evolved as a result of developments in electronic technology and the growth has been accelerated by the inability of conventional regulated exchanges to satisfy the changing needs of the marketplace. Although securities industry professionals and institutional investors generally have been the principal users of alternative trading systems, future systems will extend to all segments of the marketplace.

In many environments, private electronic crossing networks have suffered from limited liquidity because they operate on only a fraction of the total available volume in any security or other instrument. This lack of liquidity frequently results in sub-optimal execution and/or a reduced opportunity for price improvement. There is a need for an efficient e-commerce system that realizes the benefits of both internal crossing networks and external crossing networks.

Historically, program trading has been within the domain of large companies and expert traders because of the significant technical resources that are required. A basic facility for program trading involves a powerful store, expensive market connections, software for entering the market data into the store and for analyzing the stored data, and programs with wanted "conditioning" code that queries the available data and determines the appropriate reaction to specified conditions. Most retail customers do not have access to even a basic facility and many customers and companies want greater capabilities with more flexibility in reacting to market conditions than is possible with plain market orders and limit orders.

The ability to create and trade new derivatives enables companies to develop products in a tailored manner. However, derivatives introduce a new level of complexity for smaller investors that has previously only been the province of sophisticated institutional investors. Other new developments will spawn many new products that will require more flexibility, while having a capacity for greater complexity.

With new products and trading systems, there is a need to ensure that the markets are properly administered, provide accurate prices and offer adequate capacity and access, while preserving the benefits of a competitive market structure

that enhances market liquidity, transparency, anonymity and efficiency. The new products and trading systems must not compromise the need for a stable environment that minimizes market disruption.

Alternative trading systems are now regulated in the U. S. by the Securities and Exchange Commission (SEC). Under U.S. regulations, alternative trading systems can choose whether to be treated as "exchanges" or as "broker-dealers". Alternative trading systems registered with the SEC as exchanges have a need to ensure that participants comply with securities laws. Such laws may exempt from exchange regulation internal order management systems and systems that allow customers to trade solely against a dealer's inventory. Alternative trading systems registered with the SEC as broker-dealers may require, for example, regular reports, audit trails of transactions, links with a registered market, public display of quotes and orders while providing fair access to the markets.

The proliferation of new trading systems in the U.S. and throughout the world, while expanding markets, also has the effect of fragmenting the e-commerce market as each company attempts to create a proprietary system that sequesters particular market segments. These developments are rendering conventional trading systems and exchanges obsolete and creating a demand for new systems that are better able to meet the demands of the market place. Some new systems are described in the above-identified cross-referenced application.

In e-commerce systems of the type described in the above-identified cross-referenced application, there is a need for a data abstraction and data access model that enables diverse participants to share the e-commerce system while maintaining a strict division between the different participants. In addition, such a system needs to be flexible for enabling controlled data sharing between consenting participants. All of these features are needed in local as well as distributed embodiments of e-commerce systems.

Flexible systems are needed to be applicable to a wide number of institutions of different ends of the trading and e-commerce spectrum. Such systems must be parameterizable to fit the various institutions' business models and needs, while at the same time offer the high quality and reliability guarantees necessary in securities markets in all configurations. This parameterization is necessary for all aspects of the system, including crossing algorithms, crossing triggers, order eviction, and fairness bounds.

In accordance with the above background, there is a need for improved e-commerce systems that have architectures for integrating market fragments in the e-commerce market place and at the same time offer a customizable solution to crossing and processing trades efficiently.

SUMMARY

The present invention is a crossing method and apparatus for an e-commerce system for electronic transactions. Electronic transactions involve participants having interests in instruments to be crossed. A crossing unit stores variables for use in mapping the interests of the participants, stores constraints for limiting the mapping of interests of the participants and stores an expression relating the variables. The crossing unit operates to determine values of the variables that optimize the expression, while satisfying the constraints, thereby mapping the interests of the participants.

In one embodiment, the crossing unit forms an estimate of the values that optimize the expression. In one variant, processing continues at least until the estimate is reached. In another variant, the processing continues until a percentage of the estimate is reached.

In another embodiment, the crossing unit stores values of the variables for one iteration of determining values and uses the stored values in a subsequent iteration.

The present invention in one embodiment is implemented in a crossing component of a market engine where the e-commerce system for electronic transactions using external transaction units for transactions, networks for interconnecting the transaction units with one or more market engines. The market engines supervise transactions based on information gathered from the e-commerce system. The market engine includes a plurality of components for processing transactions including transaction unit interface components for interfacing with transaction units; execution components for executing transactions and a connection element connecting said components.

Each of the plurality of components includes computers, operating systems executing on the computers and application processes executing on the computers under control of the operating systems. The application processes include a function application for executing functions, a communication application for

controlling communication among the components, a resource management component for controlling the allocation of processes among the computers.

The market engine operates using execution components selected from a group that includes, for example, a routing component, a trigger component, a crossing component, a scripting component, a stock component, a bond component, a currency component, an options component, an accounting component, a storage component, and a supervisor component. The market engine operates using interface components selected from a group that includes a transaction initiator interface component, a transaction processing interface component and a data access interface component. The market engine operates using, at times, auxiliary components such as an accounting component, a storage component, a supervisor component. The partitioning of functions in different components is based on different functions can be modified to include many different combinations and other components not specifically identified can be used.

The market engine operates to supervise transactions and their routing and submission to transaction processors and thus operates to overcome the fragmentation of the e-commerce market represented by many diverse transaction processors.

In one embodiment, the market engine has a close association with a transaction processor for providing "internal" transaction processing in an integrated market engine. By having a market engine integrated with a transaction processor, internal crossings and other internal executions are performed efficiently. Further, because the market engine has knowledge of external data about similar transactions and has other external data available, the integrated market engine does not make decisions in a vacuum without reference to the entire e-commerce system. Such external knowledge about the e-commerce system is used to ensure fairness of internal transactions as measured across the entire e-commerce system. Particularly, internal crossings can be determined as fair in relationship to possible external crossings. Additionally, the internal crossings are fair in a manner that is or can be transparent to the external transaction processors. Fairness is promoted when the market engine considers information about internal transactions based upon transactions from at least one external transaction processor.

The foregoing and other objects, features and advantages of the invention will be apparent from the following detailed description in conjunction with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 depicts an e-commerce system having a group of market engines for supervising and integrating the operations of multiple transaction units.

FIG. 2 depicts details of the transaction units of FIG. 1.

FIG. 3 depicts a typical market engine formed of multiple components of the type typically used in the market engines of the e-commerce system of FIG. 1.

FIG. 4 depicts an e-commerce system having market engines and external transaction processors.

FIG. 5 depicts an e-commerce system having market engines located in different groups.

FIG. 6 depicts details of a typical component of within a typical market engine.

FIG. 7 depicts details of the interconnection of multiple components of the FIG. 6 type.

FIG. 8 depicts the logical flow for virtual ECN operations using a subset of the possible components.

FIG. 9 depicts an example of the logical flow of information from one component to another component.

FIG. 10 depicts the flow of data through the optimization process.

DETAILED DESCRIPTION

E-commerce System -- FIG. 1

FIG. 1 depicts an e-commerce system 2 for performing e-commerce transactions using electronic networks 13. Such e-commerce transactions include, for example, buying, selling, negotiation, crossing, and analysis as related to electronic instruments such as stocks and bonds, foreign currency, commodities, derivatives, books, insurance, real estate, information and any other "widget" or "entity" having value. The networks 13 include any electronic network such as the Internet, wide area networks (WAN), local area networks (LAN), public switched telephone networks (PSTN), wireless networks and any other type of public or private network and any combination thereof.

Transactions in the system of FIG. 1 are initiated, in some instances, with transaction initiators in one or more of the transaction units 7, designated as transaction units 7-1, ..., 7-Tr. Transactions are processed, in some instances, in transaction processors in one or more of the transaction units 7. The transaction initiation and processing is supervised by one or more market engines 9, designated as market engines 9-1, 9-2, ..., 9-Ma. In some embodiments, one or more of the market engines 9 are capable of initiating and processing transactions internally when one or more initiation and/or processing components are integrated into the one or more market engines 9. In other embodiments, one or more of the market engines 9 are used to unify liquidity of different transaction processors.

The market engines 9 respond to initiated transactions and supervise interaction among the transaction units 7 and the different market engines 9 to control the routing of the initiated transactions, the processing of transactions and the coordinating, gathering, storing and distributing of information useful for transaction supervision and processing.

In the FIG. 1 system, the market engines 9 are able to access and maintain information about transactions collectively as well as about each of the individual transactions being processed in the market engines 9. Where high reliability in transaction handling is required, the connections among transaction units 7 and market engines 9 are redundant or are otherwise configured to ensure high reliability and high availability.

In the FIG. 1 system, connections between transaction units 7 and market engines 9 is generically shown through networks 13 and it is to be understood that such connections in networks 13 can include direct connections between particular transaction units 7 and between particular market engines 9. Each of the market engines 9 typically receives information from all, or a significant subset, of the transaction units 7 and the other market engines 9 so that each market engine 9 is able to supervise the routing and control of transactions based upon an overview of the e-commerce system 2.

In the FIG. 1 system, the transaction units 7 each represent different transaction initiators, such as brokers and brokers" customers, or different transaction processors, such as exchanges, ATSes and ECNs. As the number of transaction units 7 increases, the greater becomes the fragmentation of the marketplace into different pools ("liquidity pools"). The e-commerce system 2, however, overcomes marketplace fragmentation by interconnecting the many

different pools of liquidity into a virtual common marketplace. The e-commerce system 2 promotes high liquidity by integrating the liquidity pools represented by the different transaction units 7 and the market engines 9. The e-commerce system 2, when connected to the world's marketplaces creates a centralized global order book. The e-commerce system 2, when connected to a company's branches creates a centralized company-wide order book. Such connections reduce overhead by streamlining transaction handling and thereby cut operational and organizational costs.

The e-commerce system 2 of FIG. 1 is flexible and accommodates a full range of financial transactions, including stocks, bonds, foreign exchange and other commercial "widgets" or "entities". System 2 provides an architecture for robust and high-speed intelligent routing and processing of transactions in a global, 24/7 marketplace with high-reliability and uninterrupted operation. Optimal executions are promoted in the e-commerce system 2 in a manner that tends to overcome the sub-optimal executions that are characteristic of fragmented financial markets. Such improvements are beneficial to all companies and customers that participate in the e-commerce system 2. Transaction processors benefit from a seamless integration, and in some embodiments are able to process transactions internally while guaranteeing fairness of executions by connecting to a set of market engines. The market engines in the FIG. 1 system operating together in a group promote consistently fair executions for customers, thereby meeting the fairness burden for each of the participants, while greatly improving information gathering and dissemination for all.

In FIG. 1, one or more of the market engines 9 is typically integrated to include transaction unit (TU) components. The integrated transaction unit components in one or more of the market engines 9, for example, include transaction initiators, such as brokers and customers, and include transaction processors, such as exchanges, alternative trading systems (ATS), data components and electronic communication networks (ECN). Many different combinations of components can be integrated into each market engine 9. The integrated market engine 9 is particularly useful for companies that wish to internally cross orders of their own customers, that is, match a company-internal buy order with a company-internal sell order of another of the company's customers. The company can also act as the principal in any or all transactions. In the disclosed system, integrating multiple transaction units is equally applicable to integrating internal transaction

processors with other internal transactions processors, internal transaction processors with external transaction processors as well as external transaction processors with other external transaction processors.

In FIG. 1, typically, the crossing components function for a company regardless of whether or not the company has branches and irrespective of the number of those branches. The integrated market engine 9 thus forms a company-wide order book internally for all the branches and potentially externally for all companies connected to the entire system 2. The internal crossing component allows a company to maintain internal control over crossing operations while significantly reducing execution costs that are associated with external crossing. These saved costs include (but are not limited to) routing costs, transactional costs, execution costs and commissions along with managerial and other information technology, costs.

Other benefits of the internal crossing component include access to the other components of the market engine. For example, the intelligent routing component of a market engine 9 provides an ability to monitor and access external liquidity pools at other market engines in real-time and with reliable, around-the-clock operation. Such pools can be dynamically added. Although a market engine may be partially impacted by the unavailability of particular pools, a market engine continues to serve and utilize the remaining pools.

Transaction Units -- FIG. 2

In FIG. 2, transaction units 7 include, for example, transaction initiators 10, such as brokers 20 and customers 23 (including funds and retail customers), and include transaction processors 12 such as exchanges 24, alternative trading systems (ATSes) 26, data units 28 and electronic communication networks (ECNs) 25. Transaction initiators are not restricted to just buy-side (money managers such as pension funds, mutual funds and hedge funds) or sell-side institutions (brokers that buy and sell on behalf of others) but include any entity capable of initiating a transaction. Similarly, transaction processors include any entity capable of processing one or more transactions. This includes transaction aggregators, or any other entity modifying or using the information contained in the initiated transaction for processing.

Market Engine with Multiple Components -- FIG. 3

In FIG. 3, one typical market engine 9 is shown formed of multiple components 71, designated as components 71-1, ..., 71-Co interconnected internally by a connection element 67. Each of the components 71 of FIG. 3 includes a computer that can range in size from small computers, such as standard personal computers (PCs), to large-scale computers, such as mainframes. Single and multi-processor computers and single and multi-computer configurations can be employed for each of the one or more components 71. Also, two or more components 71 can reside on a single hardware platform formed of one or more computers. Such two or more components on a single hardware platform are considered as logical components that are the equivalent of the physical components that exist when each component resides on a separate hardware platform. One or more of the multiple components 71-1, ..., 71-CP connects to the network 13 of FIG. 1 whereby market engine 9 connects to the transaction units 7 and to other market engines as indicated in FIG. 1.

The market engine 9 of FIG. 3 is based upon a distributed component architecture whereby the components 71, including components 71-1, 71-2, ..., 71-Co, are interconnected by connection element 67. The connection element 67 is a logical entity that provides the necessary physical interconnection of each of the components 71. The connection element is a local area network, addressable spaces in a storage component, a point-to-point switch and using any media including wireline, wireless such as infrared and microwave or any other means of transferring data between components. Regardless of the physical implementation of element 67, it functions to connect both homogeneous as well as heterogeneous components with logical consistency in the transfer of data.

In FIG. 3, the components 71 include, for example, a routing component, a trigger component, a crossing component, a scripting component, a stock component, a bond component, a currency component, an options component, an accounting component, a TI interface component, a TP interface component, a DA interface component, a storage component, a supervisor component and other components.

Routing Component. The routing component functions to route transactions and process information external to market engine 9. The routing component processes routing instructions and also potentially receives inputs from or provides outputs to the trigger component, the stock component and similarly other components. The routing component is characterized as an execution

component in that it controls remote execution. The routing component, because it actually executes trades externally is not merely a component limited to displaying a list of comparable external prices. In making decision about execution of external trades, the routing component also uses information about one or more internal orders and tries to find an optimal execution constellation considering both the internal and external information. The routing component is particularly suitable for an institution with many trades. With such internal and external consideration of information, better prices for trades are achieved. Although the operation can be limited to aggregation of internal orders, such as is done by Priceline.com for some particular e-commerce items, for example, the routing component greatly enhances the operation by also considering external markets. The consideration of the combination internal markets for efficiency and the external markets for fairness and higher liquidity provides substantial benefit both to the market engine owners and to the customers.

Crossing Component. The crossing component functions to internally cross orders available internally to the market engine 9 and hence functions as an internal ECN. In one example, when the market engine 9 is in a market engine owned by a single company, the company can cross orders of its own customers, that is, match a buy order from one of the company's customers with a sell order of another of the company's customers. In the case of a single company, a company-wide order book is maintained in a storage component 71. When the market engine 9 is in a market engine shared by a number of companies, then each company can cross orders of its own customers, that is, match a buy order from one of the company's customers with a sell order of another of the company's customers. In the case of multiple companies, each company has its company-wide order book maintained in a storage component in a secure manner that is confidential and not available to other companies unless access to other companies has been granted.

The crossing component 71-3 has customizable parameters that include, for example, crossing triggers, price determination algorithms and crossing behaviors on a per symbol basis or eviction line or increment on a per order basis. The crossing component is capable of ensuring fair execution with respect to internal as well as external data references. The internal crossing by crossing component offers a fast, expedient, and cost-effective order execution mechanism, particularly for companies with large trading volumes. The internal crossing component 71-3,

by crossing matching orders internally, avoids the typical execution costs associated with crossing in external exchanges.

The internal crossing component does not suffer from limited liquidity because it is not limited to only the internal volume that is typically only a fraction of the total available volume in any security or other instrument. Since a lack of liquidity frequently results in sub-optimal execution and/or a reduced opportunity for price improvement, the e-commerce system 2 in FIG. 1 overcomes these problems by making available both internal and external crossing. The benefits of internal crossing are, in some embodiments, conditioned upon being within a threshold of externally available executions as determined, for example, using the stock component to search for available external crossings. By use of the combination of the internal crossing with the stock component 71, optimal execution is promoted. Furthermore, exact cost calculations can be applied before routing a trade, taking such things as routing costs, execution costs, order flow into account.

In cases where an internal execution is not possible or would be unprofitable, the stock component redirects the order using the routing component to a more nearly optimal external destination for crossing. The stock component also provides an opportunity to execute market orders on a crossing network by treating them as limit orders with the limit constantly adjusted to the best external quote or by immediately executing them at the currently best price internally or externally.

The customization features of the internal crossing component include, for example, control over how orders are crossed, how the crossing price is determined, and the properties of the crossing trigger based on such parameters as time window, volume, number of orders present.

With an internal crossing component, the market engine owner has control over transactions and can decide how to exercise that control. Optimization is an example of how control can be exercised. The market engine owner can cross, for example, to maximize the spread or maximize the volume of orders being executed internally. If internal orders are less expensive to execute, there is an incentive to maximize internal orders. If internal crossing is about the same cost as external, then a company might choose to cross internally or externally in a manner that optimizes volume. For example, aggressively priced orders might be selected for crossing internally and thereby increase volume relative to what is likely if the

orders are placed last in line to execute on an external ECN (using a first-in-first-out algorithm).

The properties of optimization are not as easily controlled with external ECNs as they are with internal crossing components. When optimizing a variable, it is generally requires adversely impacting one order for the benefit of one or more other orders. While a first company, when doing an internal crossing, might not object to impacting one of its orders for the benefit of other of its orders, the first company, when doing an external crossing, will likely object if the adverse impact is to one of its orders and the benefit is to some other company's orders. Optimization across multiple companies is difficult since each company has competing interests that are difficult to reconcile instantaneously. Furthermore, external ECNs tend to keep their order books secret so that adverse impacts on some orders for the benefit of others is not readily discernable. While joint agreements between ECNs and other companies might provide for better external optimization, an impartial scheme operational among external ECNs does not seem likely.

In a multi-participant environment, optimization is generally "unfair" to some participant. Participants in control of the crossing, however, have the ability to define fairness. A company owning a market engine with an internal crossing component that is coupled to external information can insure that internal crossings for that company are always made fair relative to crossings on external markets. A company is free to set its own policies relative to internal crossings and can make them at least as fair as crossings available on external markets. Over time, fairness is promoted even though only some participants benefit in some crossings while other participants benefit in other crossings. The net benefit when averaged over a period of time can be improved even though variances in benefits.

In a multi-company environment with each company having an internal crossing component, the companies can share crossing opportunities in a sequential manner. For example, a first company's orders first go to the first company's internal crossing component and then, if not crossed, they go to a shared second crossing component operated together with a second company. If they are not matched in the second crossing component, they go to a third crossing component shared by a still different third group of companies. Every step in the sequence from the first, to the second and to the third crossing components of the first company's order increases the access to more of the other companies" liquidity, but

likely reduces the first company's ability to more favorably promote the first companies orders relative to the orders of the second and third group of companies.

Each company having an internal crossing component can select which external market (or which sequence of external markets) has preference for any order that cannot be crossed internally. Also, the sequencing of orders not crossed internally can be controlled for external submission. The external orders can be rearranged for external placement in a manner that is different than for internal crossing. Such optimization can also be used in an environment where legal or other requirements force an institution to send out a portion of their liquidity to an external transaction processor. In such an environment, the crossing engine can be used to determine a constellation for which internal use of liquidity is optimal, given the constraint that at least a specified percentage of orders are sent out externally. In effect, the crossing algorithms then determines which orders are sent externally."

Stock Component. The stock component includes stock algorithms running in the market engine 9 of one or more market engines. The stock component determines near optimal execution among many of the internal and external liquidity pools which are connected in the e-commerce system 2. The stock algorithms are used, for example, in conjunction with an internal crossing component which can be an instance of the crossing component 71, or which can be a crossing component incorporated directly in the stock component 71, to provide pricing references for determining both internal and external crossing potentials. Alternatively, the stock algorithms are used standalone to route orders to nearly optimal external execution destinations for remote executions. In a simple implementation, a stock shopping product is used to determine the best combination of publicly-visible orders from open books. More elaborate implementations are possible including finding an optimal execution involving splitting an order, pooling an order with other orders routed by the market engine and using other order scheduling optimization techniques.

An example algorithm of a stock component involves a multiple-submission technique for examining closed books or reserve orders. The technique involves the submission and withdrawal of an order at a number of closed-book markets, in a round-robin fashion. The stock component operates quickly and creates an illusion of parallel submission to all destinations. If the order is

submitted at a limit that is more desirable than the best quote available from open books, such a round-robin submission can potentially achieve a better execution than any visible quote.

Bond component. The bond component operates to negotiate trades of corporate and municipal bonds electronically. The bond component connects to broker/dealers, fund managers, banks and other locations in the e-commerce system 2 of FIG. 1. The bond component matches to portfolios presented to the bond component and the portfolio can be within the same or a different company. The bond component periodically and continuously analyzes bonds in available portfolios and attempts to satisfy as many buy and sell bond orders as possible. The bond component operates to minimize the total transaction costs (which typically result from large spreads). The bond component need not show a preference for any individual investor and can provide fairness across the e-commerce system.

The bond component is, in one embodiment, targeted to the municipal and corporate bond markets. Since the intrinsic value of a bond is generally perceived to track the going market price, market participants are typically amenable to trading bonds in their whenever spreads in the marketplace are reduced.

The bond component uses "bartering" to reduce the spread on bonds and hence is a promoter of bond transactions. The bond component operates, by way of example, with a first user posting an available bond profile including a sell list, having the bonds in the portfolio that the user is willing to sell, and a buy list, having the bonds that the first user is interested in buying. If at any point, there is another user that wants exactly the opposite transaction (buy and sell interests reversed), the users exchange the bonds directly, eliminating any cash step, where possible. Such an atomic barter transaction saves both participants the spread on two conventional transactions. The exchange rate can be determined based on the price relative to treasuries if the bonds in question have the same maturity. If the users involved in the transaction are willing to exchange bonds of mismatching maturities, then a common duration measure is applied.

The bond component also provides a source to purchase or sell municipal and corporate bonds for cash and therefore provides the users with a source of liquidity.

Currency Component. The currency component operates to buy and sell different currencies. Currency is another form of an instrument and operates in many ways like the buying and selling of any other instrument. The currency

component can operate to find the most favorable exchange rate by connecting to various external currency exchanges or can be connected to the currency exchange desk of a major institution.

Options Component. The option component operates to buy and sell different options. Options are another form of an instrument and operate in many ways like the buying and selling of any other instrument.

Accounting Component. The accounting component operates to perform the accounting functions of the market engine 9 and includes keeping track of transactions, reporting and billing among other accounting functions, usually be integrating to the backend system of the operator.

TI Interface Component. The TI interface component includes gateways and other elements to provide protocol and other conversions that are necessary between the market engine 9 and the transaction initiators 10. An example of another conversion is to accept different representations of orders (with different formatting, order of fields, etc.) and perform a consolidation step in which the orders are converted into a single representation. After the crossing, orders can then be converted back either to the old format or can be left in the uniform representation for consistency. This is particularly useful when integrating crossing into a legacy system. Typical interfaces for transaction initiators 10 are FIX and FIXML.

TP Interface Component. The TP interface component includes gateways and other elements to provide protocol and other conversions that are necessary between the market engine 9 and the transaction processors 12. Typical interfaces for transaction processors 12 are FIX and FIXML. Many other proprietary or non-proprietary protocols are possible. For example, the island ECN uses the proprietary OUCH protocol.

DA Interface Component. The DA interface component includes gateways and other elements to provide protocol and other conversions that are necessary between the market engine 9 and external data sources.

Storage Component. The storage component provides storage for the market engine 9. The storage component can be partitioned into an instruction store for storage of instructions and a data store for storage of data or can be a single common storage component. The storage component can be used to offer a global view of all trades of an organization or parts of the e-course system 2.

Supervisor Component. The supervisor component functions to operate when any of the components of FIG. 7 are operating in an abnormal manner or is experiencing other problems.

Other Components. Any number of other components can be included in the FIG. 3 market engine. A streaming compound for the streaming distribution of information collected by the market engine in one example of another component.

E-commerce System -- FIG. 4

FIG. 4 depicts an e-commerce system 2 of the FIG. 1 type for performing e-commerce transactions connected by electronic networks 13. Transactions in the system of FIG. 4 are initiated, in some instances, with external transaction initiators 10' in a typical transaction unit, designated as transaction initiators 10'-1, ..., 10'-TI. Transactions are processed, in some instances, in external transaction processors 12. The transaction initiation and processing is supervised by one or more market engines 9, designated as market engines 9-1, ..., 9E. In some embodiments, one or more of the market engines 9 are capable of initiating and processing transactions internally and are then characterized as integrated market engines.

In FIG. 4, the transaction initiators 10 are, for example, users that include computers, terminals and other equipment and software useful for persons (individuals or companies) to electronically connect to an e-commerce system. Alternatively, the transaction initiators may be brokers or other buy- or sell-side institutions including funds, investment banks among others. Brokers include computers, terminals and other equipment and software useful for persons (individuals or companies) acting as brokers for users to electronically connect to an e-commerce system. The transaction initiators in FIG. 4 can be of the user-only type of transaction initiator, can be of the broker-user type of transaction initiator or can be of any other type. Any number of such transaction initiators 10 of different types can be used in an electronic system of FIG. 4 for initiating electronic transactions. As additional examples, hierarchies of brokers, funds, institutions and users are included such as broker-broker, user-user, broker-broker-user-user, and user-fund-broker. A hierarchy in any depth or configuration can exist.

The market engines 9 respond to initiated transactions and supervise interaction among the transaction initiators 10, the transaction processors 12 and the different market engines 9 to control the routing of the initiated transactions,

the processing of transactions and the coordinating, gathering, storing and distributing of information useful for transaction supervision and processing. In some embodiments, historical data is used in this routing process to take advantage of statistical patterns in the processing performed in external transaction processors. Such historical data includes execution price and depth of the market among other things.

In the FIG. 4 system, the market engines 9 are able to access and maintain information about transactions collectively as well as about each of the individual transactions being processed in the market engines 9. Where high reliability in transaction handling is required, the connections among transaction units 7 and market engines 9 are redundant or are otherwise configured to ensure high reliability and high availability.

In the FIG. 4 system, connections among the transaction initiators 10, the transaction processors 12 and the different market engines is generically shown through networks 13, but it is to be understood that such connections in networks 13 can include direct connections among the transaction initiators 10, the transaction processors 12 and the different market engines 9. Each of the market engines 9 typically receives information from all, or a significant subset, of the transaction initiators 10 and the other market engines 9 so that each market engine 9 is able to supervise the routing and control of transactions based upon an overview of the e-commerce system 2.

In FIG. 4, the transaction processors 12 include one or more conventional (or non-conventional) exchanges 24. The exchanges include, for example, conventional exchanges 24-1, ..., 24-EX, which are, for example, the New York Stock Exchange (NYSE), Chicago Mercantile Exchange, National Association of Securities Dealers Automated Quotation System (NASDAQ), and other similar exchanges. In the FIG. 4 embodiment, the transaction processors include the alternative trading systems (ATS) and particularly, ATS 26-1, ..., 26-AT. The transaction processors also include electronic communication networks (ECN) including the ECN 25-1, ..., 25-EC. Any number of other transaction processors 27 are possible in the transaction processors 12 of FIG. 4, and these are generically indicated as the other transaction processors 27-1, ..., 27-OT. Other transaction processors include, for example, clearing houses, data aggregators, and bulletin boards. Some of the transaction processors 12 in FIG. 4 include data components for receiving or providing data relevant to transactions and these data components

are designated as the data components 28-1, ..., 28-DA. Such data components typically provide information about one or more of the other transaction processors such as the exchanges 24, ECNs 25 and the ATSes but also can provide any other type of data such as weather data, company earnings, political and economic data and so forth. Also, the data components may store data, provide data for quotations and otherwise act in any capacity to serve or receive data of all types.

In FIG. 4, the market engines 9 can include a broker for servicing, for example, any of the transaction initiators 10 whereby the broker or any other buy- or sell-side institution is "internal" to the market engine. Alternatively, the brokers can be fully independent of the market engines. In a similar manner, any one or more of the transaction processors 12 of FIG. 4 can also be closely associated with a market engine 9 and hence be "internal" to the market engine 9. The particular internal or external configuration of the different transaction initiators 10, including brokers or users, and the different transaction processors 12 with a market engine 9 is a matter of design choice. Generally, "internal" components work efficiently together under common control without need for gateways or other protocol converters. Regardless of the configuration, the market engines are connected in common among all of the transaction initiators 10 and the transaction processors 12 that constitute the electronic system 2.

By having the common connection of the market engine to the transaction initiators 10 and transaction processors 12, supervision of all similar transactions that are processed in the electronic system 2 is possible. Such supervision can be used to ensure fairness of transactions across the entire commerce system 2. Fairness is promoted when a market engine 9 has access to and considers information based upon transactions from multiple transaction processors. This could happen in collaboration with multiple transaction processors or simply by connecting to multiple such transaction processors. Fairness can also be achieved by a syndicate of transaction initiators sharing a market engine group. For example, a syndicate of funds might join their liquidity to entirely bypass the brokerage firms.

In FIG. 4, the functional flow of information is shown by broken lines, while physical connections of the transaction initiators 10, market engines 9 and transaction processors 12 are generally through direct connections to the network 13 as shown by solid lines.

Market Engine Groups -- FIG. 5

In FIG. 5, a plurality of market engines 9 are distributed in different groups 51 including groups 51-1, 51-2, ... 51-3, 51-G and connect through the networks 13 to form an e-commerce system 2. The groups 51 are organized on geographical, company, types of instruments processed (such as stocks, bonds, quotes or information) or other logical basis.

In one geographic example, the group 51-1 includes market engines 9 located in Europe. Group 51-2, by way of example, includes market engines 9 located in different countries in Asia. Group 51-3, for example, includes market engines 9 located in eastern United States and group 51-G, by way of example, includes market engines 9 located in western United States.

In the described geographic example, the FIG. 5 worldwide e-commerce system 2 is controlled in different ways. In one control example, each of the regions is a predominate region for controlling transactions during the principal business hours of that region thereby implementing follow-the-sun operation. Since the principal business hours change as a function of time and location around the world, transactions that are principal at one point in time in group 51-1 are shifted to ones of the other regions 51 at different times of day relative to a common timing source or on demand for any particular orders.

In order to control the operations of a group of market engines, each market engine 9 within a group includes a component for controlling the group operation. In the present geographic example, the group control is centered, for example, in routing components, one in each market engine, whereby transactions are routed to different market engines in different groups as a function of time or as a function of other parameters. Each routing component in each particular market engine includes a communication process that controls the operation of that particular market engine in accordance with operations common to said group. In this fashion, any desired hierarchy or pier-to-pier configuration can be achieved for each group and even across different groups.

In a company example, the FIG. 5 worldwide e-commerce system 2 has each of the groups 51 representing different constellations of market engines 9 for a single company or a group of companies. For example, group 51-1 includes all of the market engines 9 in a company that service one geographic region (for example, Berlin) while group 51-2 includes all of the market engines 9 in the company that service another geographic region (for example, New York). In such

a company example, a component for controlling the group operation is the routing component. In this embodiment, a company basically operates in multiple local hubs that are connected by a global bus. Filtering is applied when sharing information between such hubs to prevent information to be accessed across hubs that is limited to the local jurisdiction.

In another company example, the FIG. 5 worldwide e-commerce system 2 has each of the groups 51 representing different constellations of market engines 9 for a single company or a group of companies. For example, group 51-1 includes all of the market engines 9 in a company that service a particular type of instrument (for example, stocks) while group 51-2 includes all of the market engines 9 in the company that service another type of instrument (for example, currency). In such a company example, components for controlling the group operation include both a routing component and a stock component for group 51-1 and include both a routing component and a currency component for group 51-2.

The above examples are illustrative and any combination of components can be used to establish the common control functions within a group and within the e-commerce system.

Extendable Component Design. -- FIG. 6

In FIG. 6, a typical one of the components 71 of FIG. 3 is shown. The component 71 includes one or more computers 43-1, ..., 43-Ha, each computer having an operating system (OS) 42-1, ..., 42-Os, respectively. Application processes 41 are distributed to execute on the computers and operating systems 43 and 42 under control of a resource management process 46 for load balancing, fault tolerant operation and other management processes. Communications from and to the component 71 of FIG. 6 and the other components in market engine 9 of FIG. 3 are under control of the communication process 45. The particular one or more functions of the component 71 of FIG. 3 (for example, the routing function, F1, the crossing function, F2, or any of the other functions (F3, ..., F_{Co}) of the other components 71-3, ..., 71-Co of FIG. 3) are under control of the functions 44 of FIG. 6. The components 71 can be single function or multiple function depending on the capacities of the computers 43 and the overall processing load. The communication processes 45 control the communication with other components of FIG. 3 over the connection element 67 and control the communication external to the components of FIG. 3 over the connection element 68.

Extendable Component Connections -- FIG. 7

In FIG. 7, the components 71-1, ..., 71-Co are representative of the components 71 of FIG. 3 and FIG. 7 interconnected by the connection element 67. By way of example, the component 71-1 is shown having a single function, such as a crossing function, while the components 71-Co have multiple functions as shown in FIG. 6. The communication processes 45 control the communication among the components of FIG. 7 over the connection element 67 and control the communication external TO the components of FIG. 3 over the connection element 68.

The communication processes are ones that are suitable for the connection element 67. Since the connection element 67 is a logical entity that provides the necessary physical interconnection of each of the components 71, the actual processes included in the communication processes 45 are selected to satisfy the types of physical connections implemented. When the connection element is a local area network, for example, communication processes 45 provide for IP address assignment and addressing as a means to control communication among the components 71. When the connection element uses point-to-point switching, for example, communication processes 45 provide for point-to-point switching connections for transferring data between components. Regardless of the physical implementation of element 67, the communication process 45 have a logically consistent interface between components 71 that permits both homogeneous (for example, using the same hardware computer and operating system) as well as heterogeneous (for example, using different hardware computers and operating systems) components to transfer data.

The communication process 45 controls communications over the communication element 67. Depending on the implementation of the communication element 67, the communication process 45 uses different mechanisms to communicate with the communication processes 45 of other components. In one particular embodiment, the communication process 45 uses object serialization to transmit message (or other) objects from one communication process 45 to another. This operation is done by initiating a network connection (for example a TCP/IP connection), then serializing the message object into a data stream (which is usually buffered). The data stream is then transmitted by the transmitting communication process 45 over the TCP/IP connection unity 67 is received by the receiving communication process 45 and is de-serialized at the

receiving communication process 45. For example, one embodiment sends meta-data of a buy/sell order from an interface component to a storage component and subsequently to the crossing component. The Java Remote Method Invocation (RMI) interface by Sun Microsystems can be used to implement such communication methods via interchange of serialized objects.

For different message-types and embodiments of the connection element 67, the use of other communication protocols with different flow-control mechanisms, delivery guarantees and directory services can be used. Various schemes over IP are possible. For example, heartbeat messages use the UDP/IP protocol because reliable delivery is not required. Communication protocols are not restricted to IP-based schemes, given that both the transmission component as well as the receiving component are capable of handling messages in a selected format. Specialized fast messaging systems, Remote Procedure Call (RPC) and Active Messages are acceptable implementations as well. In other embodiments, higher-level messaging systems are used to communicate between components. Examples include TIBCO, NEON or VITRIA messaging layers which are again able to completely abstract the communication layer from the underlying hardware components and thus effectively act as middle-ware.

In further embodiments, multiple components run on the same hardware and operating system node using the same memory. In this embodiment, the same communication mechanisms can be used as described above. Additionally, however, specialized inter-process communication schemes are possible for improved performance and better use of system resources.

Market Engine Crossing Operation -- FIG. 8 - FIG. 9:

FIG. 8 and FIG. 9 illustrate the distributed component system architecture of market engine 9 using, by way of example, a group of six components 71. The six components are specifically supervisor component 71-12, TI interface component 71-10, TP interface component 71-11, crossing component 71-3, storage component 71-13 and routing component 71-1. In FIG. 8, a global order book, stored in the storage component 71-13, interacts with a transaction initiator 10 through the TI interface component 71-10, then the trade is routed to the crossing component 71-3. In FIG. 9, the interplay between crossing component 71-3 routing component 71-1 is shown. This interaction is used to perform crossings with external reference, as well as to route trades to external transaction processors.

The data flow diagrams in FIG. 8 and FIG. 9 illustrate one-way interaction during normal operation. It is understood that the same interaction can occur in reverse.

In FIG. 8 and FIG. 9, the supervisor component 71-12 is responsible for system supervision and data flow control. This component serves to gracefully shut down components in case of network failure and aids in wide-area fail-over and other administrative tasks that cannot be regularly performed because of the failure. A microwave, satellite or other external connection 68-12 plugged into one or more workstation computers 81 provides this functionality.

In FIG. 8 and FIG. 9, the TI interface component 71-10 provides an interface to transaction initiators 10 as shown in FIG. 4, for example. In one embodiment, TI interface component 71-10 employs a loosely coupled cluster 82 of Sun Sparc workstation computers (computers 43-1, ..., 43-Ha of FIG. 6) running Solaris operating systems (operating systems 42-1, ..., 42-Os of FIG. 6). In this embodiment, incoming requests are distributed among front-end processes running on one or more of the nodes (where each hardware 43-1, ..., 43-Ha and corresponding operating systems 42-1, ..., 42-Os pair of FIG. 6 is a node) of cluster 82 in a round-robin fashion. This distribution provides availability because, even in the case of failure of one node, other of the nodes can handle a request. In another embodiment, each of the nodes also acts as a firewall to provide data security. In this embodiment, all requests are logged into a store 83 for analysis. One important task of the TI Interface 71-10 is protocol conversion between a transaction initiator 10 protocol and the market engine 9 internal protocol. Conversions are typically from FIX/ML, NEON, TIBCO, VITRIA and similar messaging layers or protocols.

In FIG. 8 and FIG. 9, the TP Interface 71-11 is similar to the TI Interface 71-10 and it is responsible for protocol conversion and proper interaction with exchanges and other transaction processors 12. A loosely coupled cluster of workstation computers 84 is employed and interactions are logged into the store 85.

In FIG. 8 and FIG. 9, the crossing component 71-3 includes one or more working clusters 86 and supervisory clusters 87. In one embodiment, the crossing component 71-3 is built as a tightly coupled cluster of Sun workstations (computers 43-1, ..., 43-Ha of FIG. 6) running Solaris operating systems (operating systems 42-1, ..., 42-Os of FIG. 6) where the nodes (each computer 43-1, ..., 43-Ha and corresponding operating systems 42-1, ..., 42-Os pair of FIG. 6 is a node) of cluster 86 are operated as peers. In one embodiment, each node is responsible for a set of

symbols to cross (for example, IBM, Intel, Yahoo). While there are thousands of symbols in the various exchanges, the cluster consists of significantly fewer nodes with each node servicing a plurality of symbols. Designated one of the nodes act as load balancers for assigning symbols to nodes. Depending on the number of trades to be crossed for a particular symbol, the balancing nodes dynamically reassign symbols to different nodes. In a case of a very active symbol (for example, trades of Yahoo on certain days) a single node is responsible for a single symbol or the processing of a single symbol can further be broken up into multiple nodes (distributed or parallel processing) if needed.

In the crossing component 71-3, the supervisory cluster 87 has processes that monitor heartbeats, register failures and are responsible for bringing up nodes after scheduled or unscheduled downtime. Other processes monitor heartbeats of the supervisor. Should the supervisor fail, the first node to detect the failure brings up a new supervisor after a random delay (possibly with exponential back off). In the unlikely event that two or more supervisors are started, each supervisor detects the presence of the other supervisors and all but the supervisor with the highest priority (for example, in a LAN embodiment, one with the highest IP address and highest process ID) commits suicide. A hierarchy of such monitors is possible for local as well as multi-site fault-tolerance.

The routing component 71-1 includes cluster 84 and store 85 and operates to provide a view of external executions and uses the view information to optimally route trades to external markets. The TP interface 71-11 provides an interface between the routing component 71-1 and the external transaction processors 12 (see FIG. 4). In one implementation, the granularity of operations is on a per security basis. This granularity is provided to facilitate optimizations (like pooling of orders) that need information about all available trades of a given symbol and the crossing component 71-3. The routing component 71-1 has the same basic structure as other components and has designated nodes running reliability processes to enhance reliability.

The storage component 71-13 functions as a storage and message handling system. In one embodiment, component 71-13 is built upon a buddy system cluster 88 of Sun Enterprise server computers running Solaris operating systems operating in a SMP (Symmetric Multiprocessing) mode. These SMP nodes of cluster 88 are responsible for maintaining a reliable storage system. In another embodiment, a group of symbols is handled on one or more (possibly dedicated) nodes. While

other components can completely fail without major consequences (jobs are simply re-run), the storage component 71-13 typically is implemented with high reliability and availability including mechanisms for local and wide-area fail-over. In such embodiments, transactions are stored both locally on store 89 as well as remotely, at a remote location over connection element 68-13, for immediate wide-area fail-over. Synchronization mechanisms 92 on a transaction level or a disk level provide a wide range of processes and data exchanges for reliability.

Overall Operation -- FIG. 8 and FIG. 9. The FIG. 8 and FIG. 9 a group components, a subset of the components of FIG. 3, form a market engine 9. The processes in the components of FIG. 8 and FIG. 9 are parameterizable, extendable, highly scaleable and fault tolerant. "Parameterizable" in this context means, for example with reference to a crossing component parameters, that different parameters such as crossing algorithms, crossing windows and order eviction policies, can be selected on a per-component, per-symbol or other basis. This operation allows the market engine 9 to customize procedures while avoiding the need for custom programming. "Extendable" refers to the ease at which algorithms or other features can be added or replaced. "scalable" denotes the ease by which multiples of a particular workload can be accommodated by the architecture. The architecture scales to each symbol, to an entire node and further to any number of nodes. "Fault tolerant" refers to the ability to detect and circumvent failures as they occur. Processes that are not successfully executed must rerun with the same parameters but potentially on a different node.

Storage Component Operation (A1-A5) -- FIG. 8. Whenever a transaction initiator 10 (see FIG. 4) enters an order (or other parameters), the order gets stored into a Global Order Book (storage component 71-13). The storage component 71-13 provides the reliable storage necessary to ensure no loss of orders --it preserves the hard state. Once an order is stored, attributes of an order can be queried and updated by a transaction initiator 10. In addition, the storage component 71-13 implements order handling operations whereby transaction initiators 10 can manually control movement of a trade to another component over connection element 67 (for example, move to the crossing component 71-3 for internal execution) or, alternatively, a trade can be assigned to an standard automated order handling routine.

An automated order handling routine supervises a particular automated sequence of submissions to the various components 71. For example, in one order handling routine, all retail trades of an institution are first sent to the internal crossing component 71-3 for internal crossing, followed by a submission of uncrossed orders to the best external destination via routing component 71-1. In another order handling routine, an institution exercises control over institutional trades and directs that they be submitted to a particular exchange 24 (see FIG. 4). The storage component 71-13 provides such flexibility.

The order handling routine illustrated in FIG. 8 starts with a transaction initiator 10 submitting an order/request through the front-end network connection element 68-10 at A1. One of the TI interface component 71-10 nodes in cluster 82 at A2 then picks up the order/request, verifies the source, logs it to a store 83 at A3 and converts the format to an internal format and sends it off as a message over the connection element 67 at A4. The order/request is then transmitted to the storage component 71-13 and is received at A5 by one of the nodes of cluster 88. The storage component 71-13 checks for consistency and conformity and then stores the order/request into reliable storage, both locally in store 89 and over remote connection element 68-13 to a redundant remote store (not shown). The synchronization procedures are subject to parameter settings. The same sequence is evoked in reverse for a response from the storage component 71-13 to the TI interface component 71-10.

The storage component 71-13 provides a global, enterprise-wide view (Global Order Book) of a financial institution's data, including status and ownership information. Component 71-13 is designed as a data store with both local and wide-area redundancy and provides follow-the-sun capabilities. It is used to store hard state.

The storage component 71-13 is one of the execution components and execution is performed with one or more of the features and functions of the following TABLE 71-13.

TABLE 71-13

- 1. Functionality
  - 1.1. Attributes and Ownership
    - 1.1.1. Reassigning Ownership
    - 1.1.2. Permissions
    - 1.1.3. Changing other Attributes
      - 1.1.3.1. Credit Check
  - 1.2. Actions and Responses to Queries
    - 1.2.1. Accepting an Order/Commands
    - 1.2.2. Synchronization
    - 1.2.3. Querying/View State
      - 1.2.3.1. Of a Single Order
      - 1.2.3.2. Of an Order Type
    - 1.2.4. Order Handling (automated- and manual procedures)
      - 1.2.4.1 Removal
        - 1.2.4.1.1 Cancellation of Order(s)
        - 1.2.4.1.2 Removal from Storage component 71-13
      - 1.2.4.2. Move to crossing component 71-3
      - 1.2.4.3. Move to routing component 71-1
      - 1.2.4.4. Move Directly to a Specified Exchange through component 71-11
    - 1.2.5. Streaming Distribution of State
    - 1.2.6. Security/Access Control
  - 1.3. Integration
    - 1.3.1. Application Programming Interfaces (APIs)
      - 1.3.1.1. Storage System
- 2. Retaining State
  - 2.1. Local Redundancy
    - 2.1.1. Synchronization (combination of the following)
      - 2.1.1.1. After N>0 Change(s)
      - 2.1.1.2. After X Amount of Time
      - 2.1.1.3. After Y Amount of Money/Shares

2.1.1.4. Synchronous Creation, Asynchronous Updates

2.2. Wide-area Redundancy

2.2.2. Synchronization (combination of the following)

2.2.2.1. After N>0 Change(s)

2.2.2.2. After X Amount of Time

2.2.2.3. After Y Amount of Money/Shares

2.2.2.4. Synchronous Creation, Asynchronous Updates

2.3 Data Encryption

3. Failure Modes

3.1. Failure of a Node

3.2. Failure of a Site

3.3. Failure of a different Component

3.3.1. Crossing Component 71-3 failure

3.3.2. Routing Component 71-1 failure

3.4. Failure of an External Process

3.5. Failure of the entire market engine

4. Monitoring

4.1. Orders/Data in the Storage component 71-13

4.1.1. Full Attributes

4.1.2. Current State

4.2. Component System States (individual components)

4.2.1. Parameter Settings

4.2.2. Current State within settings

4.2.3. Statistics (for Storage component 71-13: number of incoming orders and status changes)

4.3. System Status

4.3.1. Machine Workloads and Health

5. Auditing

5.1. Changes/Updates to Orders

5.2. Status Changes

Crossing Component Operation (D1-D5) -- FIG. 8. The crossing component 71-3 attempts to match orders from a transaction initiator 10 (see FIG. 4) before, while (or instead of) the orders are submitted to an external transaction processor 12 (for example, an exchange in FIG. 4). The crossing component 71-3 attempts to cross buy and sell orders (for example for stocks, bonds or any other widget) using a variety of algorithms. Moving a trade from the Global Order Book to the Crossing component 71-3 is similar to moving a trade to the routing component 71-1 or directly to the external markets. The steps A1-A5 represent order entry together with any commands, including the command to move the order to the crossing component 71-3. This procedure is manual or part of an automated order handling routine.

The order is taken from the storage 89, at D1, and moved by one of the nodes of cluster 88, at D2, to the connection element, at D3. The order is then picked up by one of the nodes of cluster 88, at D4, in the crossing component 71-3 which in turn routes the order to the node in charge of the order's symbol. Assistance is sought from the supervisor cluster 81 if the correct destination node cannot be located. The same sequence is evoked in reverse (D5-D1) for a message from the crossing component 71-3 to the storage component 71-13.

The crossing component 71-3 crosses orders with reference to external market conditions. The storage component 71-3 is one of the execution components and execution is performed, for example, with one or more of the features and functions of the following TABLE 71-3.

TABLE 71-3

1. Algorithms with Repeatable Results (combination as second 'sorting' criteria)
    1.1. Maximize Share Volume
    1.2. Maximize Number of Distinct Transaction initiator 10s
    1.3. Maximize Number of Orders
    1.4. Maximize Spread
    1.5. Custom Objective (by client)
    1.6. FIFO (Sorted by price, then arrival time; example: Instinet)
    1.7. Within Specified Threshold (combination with 1.1-1.5)
2. Crossing Trigger (any combination of the following, reset upon crossing)
    2.1. Fixed Conditions
        2.1.1. Based on Volume in the Crossing Component 71-3
        2.1.2. Based on Number of Orders in Crossing Component 71-3
        2.1.3. Based on External Market Conditions
            2.1.3.1. 'Sudden' Change in Price (to be defined)
            2.1.3.2. 'Sudden' Change in Liquidity (to be defined)
    2.2. Experience-based (adaptive) Sliding Window (any combination)
        2.2.1. Based on Volume
        2.2.2. Based on Number of Orders
        2.2.3. Based on Volatility
    2.3. Continuous (FIFO, i.e. after each arriving order)
        2.3.1. On all Orders
        2.3.2. On Market Orders only
    2.4. Set Time (Non-Sliding Window)
        2.4.1. Interval Time

2.4.2. Absolute Time(s) (with universal timer)

2.5. Immediate (allows for automated external or manual triggers)

3. Evicting Orders (per order, any combination)

3.1. At Set Time

3.1.1. Interval Time

3.1.2. Absolute Time(s) (with universal timer)

3.2. After Set Number of Crossings

3.3. Marketable Orders

3.3.1. Exempt based on Preliminary Optimization Results

3.4. Immediate (per order/all)

3.4.1. Order Cancellation

4. Monitoring

4.1. Orders in the Crossing Component

4.1.1. Full Attributes (side, quantity, limit, stop, etc)

4.1.2. Current State

4.1.3. Amount of Shares Already Filled

4.1.3.1. Orders Making up the Filled Part

4.2. Crossing Component State

4.2.1. Parameter Setting

4.2.1.1. Threshold for each Symbol

4.2.1.2. Crossing Behaviors

4.2.2. Current State within Settings (time to crossing, etc)

4.2.3. Statistics (for Crossing Component, number of orders crossed per symbol)

4.3. System Status

4.3.1. Machine Workloads and Health

5. Reporting

5.1. Crossings

5.1.1. Conditions at Crossing

5.1.2. Orders Making up each Crossing

5.1.3. Crossing Results

5.2. Audit Trail

5.2.1. Stored Data

5.2.2. Order Tracking and Lookup

5.3. Trade Confirmation

There are some important things to notice in TABLE 71-3. For one, crossing algorithms in this embodiment are pluggable. This means that then can be replaced easily and that switching between algorithms is seamless. It also enables the operator to apply different algorithms to different symbols or for different crossings. Since executions can be within a threshold, it is possible to only cross orders that can price-improve over externally available deals. Because of this, even orders that need to be executed immediately and at the price of the current market (i. e. market orders) can get executed in this engine, as long as either the crossing is performed immediately or a FIFO crossing is used for these orders with the orders waiting to get crossed (as long as the execution price is better or equal to the external markets). Another point to notice in TABLE 71-3 is that there is a possibility of combining multiple eviction and crossing window conditions, such that the crossing (or eviction) happens if either condition is met --example: cross when a certain volume is reached or a certain amount of time has passed. The granularity of eviction is at the level of an order rather than a crossing. So each order can get evicted at the desired time, unless a crossing is taking place exactly at the time of eviction. In this case, the order is evicted either beforehand or alternatively, right after the crossing. In one implementation, the crossing component is further designed in a way that allows for executions in any fraction. So orders in one fraction can get executed with orders of a different fraction and results are rounded to the next integer numerator of the lower granularity fraction. This allows executing, for example, orders denominated in 1/64 with orders denominated in 1/100 during a transitional phase.

Fairness Operation -- FIG. 9. This component system fairness operation is illustrated in FIG. 13. In order to update the fair crossing band of the crossing component 71-3 for trades of the same symbol, a node at E2, of the cluster 91 of fair crossing routing component 71-1, possibly with assistance from a supervisor node, of cluster 92 at E1, passes the quote update to the connection element 67. The quote update is then passed to one of the nodes, at E4, of the crossing component 71-3. This node then determines which node is in charge of the symbol in question. If the local cache does not contain the most up-to-date information, the supervisor, at E5, is consulted. At the next contemplated crossing is within the newly specified band the crossing component 71-3 guarantees that the order to be executed. For additional protection against potential bad quotes, other parameters

can be adjusted appropriately. In one embodiment, external order information (usually exceeding only quote information) is used throughout the optimization process. In this embodiment the optimization produces optimal results on the combination of internal and external orders and matches them accordingly.

Synergy between Component Systems. One of benefits of the distributed component system architecture of market engine 9 is the synergy that exists between components. One example is that the TP interface component 71-17 used by the routing component 71-1 is also used for direct order routing from storage component 71-13. The direct order function could also be implemented using an entirely different component (not shown). A significant function of the routing component 71-1 is the quote server operation of FIG. 9. The routing component 71-1 aggregates data from the various markets and creates a global view of the external world. Along with a streaming component (not shown), these components are used together to stream quotes to websites, for example, so that users can compare the distributed Component System quotes with standard quote streams of others.

Further, quotes are also used to provide thresholds ranges or other bands for the crossing component 71-3 to determine if internal or external crossings should be executed. The bounds can be dynamically adjusted according to the latest information about external conditions. Using these techniques, fair internal executions are established with reference to external markets.

Optimization Process -- FIG. 10.

In FIG. 10, the data flow for optimization in a crossing component 71-3 is shown. Optimizations on orders, in one embodiment, can be performed among only internal orders supplied by transaction initiators 10. In this case, external information (for example, market quotes) is used as a reference point only, if at all. In another embodiment, internal orders are optimized only against externally available orders in what is referred to as "bulk shopping". In this mode, internal orders are not matched against each other but can be split or pooled together to match external orders. In one preferred embodiment both internal as well as externally available orders are taken into account when optimization is performed.

Regardless of the particular embodiment, certain principals generally apply to order matching and optimization processes. A control process 92 in FIG. 10

controls the optimization. At the time of crossing, control 92 fetches the necessary data from the data unit 91 in communication D4-1 in FIG. 10. This data includes (meta) data about buy and sell orders as well as any external information adding value at the time of crossing. Valuable information includes, for example, externally available execution prices and market depth. In a second step, the control 92 pre-processes the data. This step includes, for example, checks for validity of the orders, integrity of other data used in the optimization process and analysis of fairness constraints among other pre-optimization procedures. Once the data is formatted according to the interface of the optimizer 97, it is then passed for optimizing in a communication D4-2 in FIG. 10. Optionally, other control parameters (such as which algorithm should be used) are set at D4-4 before the control 92 starts the optimization, by issuing the appropriate command in D4-4. This D4-4 interface is also responsible for the runtime control, including interrupting the optimizer 97 in the case of an exception condition, getting the currently best crossing, or changing the parameter settings of the optimizer 97. A mapping from buy orders onto sell orders (or vice versa), is referred to as "crossing." Once a desired mapping has been achieved by the optimizer 97, the resulting crossing in optimizer 97 is passed back to the control 92 in communication D4-9 of FIG. 10. The control 92 then performs post-optimization steps including, for example, assuring that the fairness boundaries are still valid or digitally signing the executions for security, consistency and accountability. This step potentially involves up-to-date external data from the data unit 91 that receives data over communication link 67. Finally, executions are stored in communication D4-10 and unbilled order from partial fills are re-introduced in the next crossing depending on the settings.

The optimization process can be generalized as follows:

1) Three items are passed to the optimizer, namely, variables, constraints on the variables (for example, expressed in terms of the supplied variables) and an expression of the variables. The variables are, for example, the number of shares matched between two orders, execution prices, or other vlaues relevant to optimization. Constraints are restrictions on the variables including, for example, a limit price, a maximum number of partial fills, even lot execution, all or nothing (AON) execution or other preferences. These constraints can be applied per order or across a set of orders. An example of a constraint on a set of orders is a price constraint on a basket of orders. Another constraint is a fairness threshold applied

on the execution price of each crossing if fairness is desired with respect to an external threshold. The expression is an optimizable relationship among the variables, for example, a relationship that allows maximization of volume while minimizing partial fills, a relationship that allows maximization of executed orders, or any other mathematically definable relationship useful in the crossing process.

2) The optimizer uses algorithms to approach the desired objective, while meeting the imposed constraints. An algorithm that optimizes for different objectives is shown in TABLE 1 below.

3) The optimizer returns a desired crossing. This result can be the empty set in case that no constellation of variables can meet the constraints or if not enough time was given to the optimizer to find such a constellation.

<u>Optimizer</u>

The following TABLE 1, TABLE 2 and TABLE 3 show implementations of representative optimization algorithms. These implementations are shown for reasons of elegance and simplicity and to demonstrate how a simple change in parameter can yield vastly different crossing results. The main differences in the following pseudo-code segments in TABLE 1, TABLE 2 and TABLE 3 are contained in simple checks to see whether the best result so far should be overwritten. These checks appear in the "base case" section of the code.

An advantage of the TABLE 1, TABLE 2 and TABLE 3 embodiments is that all of the algorithm debugging is performed once and adding new algorithms or adjusting existing ones is greatly simplified. Combining algorithms in priority order is easy. This combining can be done by simply adding a secondary check after the primary one. This combining is illustrated by adding an additional constraint to every algorithm so that, all other things being equal, orders are split in as few partial executions as possible to simplify the reporting (among other benefits). Another advantage of such an embodiment is that all-or-nothing (AON) orders are not treated as special cases.

The TABLE 1, TABLE 2 and TABLE 3 implementations are inefficient. In many situations, faster algorithms can be developed for specific optimization objectives by streamlining these algorithms. Since the time complexity for some optimization objectives (for example maximization of orders that are at least 50% crossed) is exponential, no feasible algorithm (with polynomial time complexity) can be developed to find an optimal solution. Instead, a brute-force or heuristic

algorithm is applied for a certain amount of time or until a solution within a certain threshold is reached. Such an implementation takes advantage of the fact that for a small number of orders (where liquidity is a problem), an optimal solution can be found in a reasonable amount of time, while for a large number of orders (where an optimal solution is not really necessary) a good solution is found quickly. In operation, the optimization can run continuously while new orders are coming in and these preliminary results are then used as good parameters for a heuristic algorithm.

In the algorithm pseudo-code in TABLE 1, TABLE 2 and TABLE 3 below, the set of best crossings so far (M) is initialized empty for the sake of clarity although it need not be empty. For example, a FIFO cross is run first, which is linear in time complexity, and then the exponential algorithm is used to improve on the result set.

TABLE 1

MAXIMIZATION OF SHARE VOLUME WHILE MINIMIZING ORDER FRAGMENTATION

This exponential algorithm maximizes the total share volume crossed while minimizing the number of times individual orders are split (partially filled) whenever possible. The secondary optimization criterion (fragmentation minimization) illustrates the way optimization criteria can be "stacked" in priority order.

1. Fundamentals

Define a set S of sell orders where every element s represents a distinct sell order s
Define a set B of buy orders where every element b represents is a distinct buy order b

2. Atomic Sets

Split all orders into suborders of atomic size
Define a set SA of atomic sell orders where every element is a tuple (sa, s) representing a distinct atomic order sa
Define a set BA of atomic buy orders where every element is a tuple (ba, b) representing a distinct atomic order ba

3. Optimization Sets

Define an empty set M where every element is a tuple (sa, ba) representing a match of a pair of distinct atomic orders sa and ba where sa is in SA and ba is in BA remember the current (empty) state of M as "optimal composition of" (M) This is optimal composition SO FAR. If the algorithm is allowed to run.to completion, the "optimal composition of" (M) indicates the optimal composition of M overall, given the same inputs S and B or one of such equally optimal compositions under given optimization criteria.

4. Algorithm Code

```
start
        call "match pairs" (SA, BA, M)
done

subroutine "match pairs" (by value SA, by value BA, by value M)
        define "base case" = empty" (SA) OR empty" (BA)
        if "base case" then
                if "number of elements" (M) >  "number of elements" ("optimal composition of" (M)) then
                        "optimal composition of" (M) = M

                                        // The composition of M has been improved according to primary
                                             // optimization criterion, maximum number of elements in M
                end if
                if "number of elements" (M) == "number of elements" ("optimal composition of" (M)) then
                                             // No improvement or deterioration under primary criterion
                                                                        // Try secondary criterion
                        if "number of distinct (s, b) subtuples" (M X S X B) <
                                "number of distinct (s, b) " ("optimal composition of" (M) X S X B)
                                then "optimal composition of" (M) = M

                                      // The composition of M has been improved according to secondary
                                      // optimization criterion, minimum number of pairs (s, b) of original
                                                                                // size orders
                        end if
                end if
                                                        // tertiary criterion and so on, if desired
        else
                                                                            // keep branching
                foreach sa in SA do
                        foreach ba in BA do
                                if ba.price => sa.price then
```

```
                        "match pairs" (SA - sa, BA - ba, M + (sa, ba))
                    else
                        "match pairs" (SA - sa, BA - ba, M)            // no deal
                    end if
                end foreach
            end foreach
        end if
    end "match pairs"
```

TABLE 2

MAXIMIZATION OF ORDERS FILLED WHILE MINIMIZING PARTIAL EXECUTIONS

This exponential algorithm maximizes the total number of orders filled, at least partially, while minimizing the number of times individual orders are split (partially filled) whenever possible. Number of distinct clients (order owners) can be maximized in analogous fashion, so no separate code is provided. The secondary optimization criteriaon (fragmentation minimization) illustrates the way optimization criteria can be 'stacked" in priority order.

1. Fundamentals

Define a set S of sell orders where every element s represents a distinct sell order s

Define a set B of buy orders where every element b represents is a distinct buy order b

2. Atomic Sets

Split all orders into suborders of atomic size

Define a set SA of atomic sell orders where every element is a tuple (sa, s) representing a distinct atomic order sa

Define a set BA of atomic buy orders where every element is a tuple (ba, b) representing a distinct atomic order ba

3. Optimization Sets

Define an empty set M where every element is a tuple (sa, ba) representing a match of a pair of distinct atomic orders sa and ba where sa is in SA and ba is in BA remember the current (empty) state of M as "optimal composition of" (M) This is optimal composition SO FAR. If the algorithm is allowed to run to completion, the "optimal composition of" (M) indicates the optimal composition of M overall, given the same inputs S and B or one of such equally optimal compositions under given optimization criteria.

4. Algorithm Code

```
start
        call "match pairs" (SA, BA, M)
done

subroutine "match pairs" (by value SA, by value BA, by value M)
        define "base case" = empty" (SA) OR empty" (BA)
        if "base case" then
                if "number of distinct s" (M X S X B) + "number of distinct b" (M X S X B) >
                        "number of distinct s" ("optimal composition of" (M) X S X B) +
                        "number of distinct b" ("optimal composition of" (M) X S X B)
                then
                        "optimal composition of" (M) = M

                                        // The composition of M has been improved according to primary
                                // optimization criterion, maximum number of original size orders in M
                end if

                if "number of distinct s" (M X S X B) + "number of distinct b" (M X S X B) ==
                        "number of distinct s" ("optimal composition of" (M) X S X B) +
                        "number of distinct b" ("optimal composition of" (M) X S X B)
                then

                                                // No improvement or deterioration under primary criterion
                                                                        // Try secondary criterion
                        if "number of distinct (s, b) subtuples" (M X S X B) <
                                "number of distinct (s, b) " ("optimal composition of" (M) X S X B)
                        then
                                "optimal composition of" (M) = M

                                        // The composition of M has been improved according to secondary
```

```
                                        // optimization criterion, minimum number of pairs (s, b) of original
                                                                                           // size orders
                  end if
            end if
                                                                  // tertiary criterion and so on, if desired
      else
                                                                                     // keep branching
            foreach sa in SA do
                  foreach ba in BA do
                        if ba.price => sa.price then
                              "match pairs" (SA - sa, BA - ba, M + (sa, ba))
                        else
                              "match pairs" (SA - sa, BA - ba, M)                  // no deal
                        end if
                  end foreach
            end foreach
      end if
end "match pairs"
```

TABLE 3

MAXIMIZATION OF THE TOTAL SPREAD GENERATED WHILE MINIMIZING FRAGMENTATION

This exponential algorithm maximizes the total bid-ask spread generated in all crossings crossed, while minimizing the number of times individual orders are split (partially filled) whenever possible. The secondary optimization criterion (fragmentation minimization) illustrates the way optimization criteria can be "stacked" in priority order.

1. Fundamentals

Define a set S of sell orders where every element s represents a distinct sell order s

Define a set B of buy orders where every element b represents is a distinct buy order b

2. Atomic Sets

Split all orders into suborders of atomic size

Define a set SA of atomic sell orders where every element is a tuple (sa, s) representing a distinct atomic order sa

Define a set BA of atomic buy orders where every element is a tuple (ba, b) representing a distinct atomic order ba

3. Optimization Sets

Define an empty set M where every element is a tuple (sa, ba) representing a match of a pair of distinct atomic orders sa and ba where sa is in SA and ba is in BA remember the current (empty) state of M as "optimal composition of" (M) This is optimal composition SO FAR. If the algorithm is allowed to run to completion, the "optimal composition of" (M) indicates the optimal composition of M overall, given the same inputs S and B or one of such equally optimal compositions under given optimization criteria.

4. Algorithm Pseudo-code

```
start
        call "match pairs" (SA, BA, M)
done

subroutine "match pairs" (by value SA, by value BA, by value M)
        define "base case" = empty" (SA) OR empty" (BA)
        if "base case" then
                if "sigma of (ba.price - sa.price)
                        foreach (sa, ba) from" (M) > "sigma of (ba.price - sa.price)
                        foreach (sa, ba) from" ("optimal composition of" (M))
                then
                        "optimal composition of" (M) = M

                                        // The composition of M has been improved according to primary
                        // optimization criterion, maximum sum of all bid-ask spreads in all pairing in M
                end if

                if "sigma of (ba.price - sa.price)
                        foreach (sa, ba) from" (M) == "sigma of (ba.price - sa.price)
                        foreach (sa, ba) from" ("optimal composition of" (M))
                then
                                        // No improvement or deterioration under primary criterion
                                                                // Try secondary criterion
                        if "number of distinct (s, b) subtuples" (M X S X B) <
                                "number of distinct (s, b) " ("optimal composition of" (M) X S X B)
                        then
                                "optimal composition of" (M) = M
                                    // The composition of M has been improved according to secondary
                                    // optimization criterion, minimum number of pairs (s, b) of original
```

```
                                                        // size orders
                end if
            end if
                                    // tertiary criterion and so on, if desired
        else
                                                        // keep branching
            foreach sa in SA do
                foreach ba in BA do
                    if ba.price => sa.price then
                        "match pairs" (SA - sa, BA - ba, M + (sa, ba))
                    else
                        "match pairs" (SA - sa, BA - ba, M)          // no deal
                    end if
                end foreach
            end foreach
        end if
    end "match pairs"
```

Market Making

The crossing methods described can be used alone. However, in certain situations many more sellers exist in a security or other instruments described than buyers or vice-versa. In these situations, crossings are difficult to establish because the available liquidity is one-sided. In these situations, the value of an entity works to establish an orderly market becomes clear. On the NYSE, this entity is called a "Specialist," while on Nasdaq, "Market Makers" have this responsibility.

A market maker is valuable in combination with a crossing engine, particularly with optimization features. Integrating a market maker into the crossing process adds additional value.

Routing orders to a market maker when no match is found during the crossing is one way of achieving this goal. In one embodiment, market makers include large blocks of a security in a crossing and specific label them as market maker blocks. During the optimization process, these blocks are then treated differently, for example, instead of maximizing for executed share volume, the applied optimization is to maximize share volume while minimizing the number of executions performed against the market maker blocks. In the case of one-sided liquidity, the market maker blocks are used. In the case of a balanced market, buyers and sellers are preferred over market makers. Market makers have an incentive to participate if they can share in the speed or can mark up the blocks.

Applications of the Crossing Component. The system described is customizable to apply to different contexts in various financial and e-commerce markets. The customizable properties can be used to tailor the system to the specific objectives of an institution. While every institution's requirements are different in the financial services industry, the following broad categories exist:

Retail brokerage – Retail brokerages are likely to maximize the number of individual orders passing through the system in order to increase pre-order commission totals or generated spread in order to maximize the order flow payments. An online retail brokerage is significantly more sensitive to response times and thus more likely to utilize a smaller time trigger for the crossing window.

Institutions managing institutional assets or trading for their own account --Such institutions are likely to maximize the total share volume executed internally in order to avoid the costs of external execution and potential market impact. In this environment, large block orders are significantly more common, so the eviction strategy for unmatched orders is likely to be engineered to consider the

potential impact on external markets. For block trades, more time can be spend optimizing orders against each other and the monetary impact of good executions is larger.

International securities dealer affiliated with a bank or other significant forex dealer -- Such institutions are likely to be interested not only in crossing mutually offsetting orders for foreign securities internally, but also cross the orders for the required amounts of the denominated foreign currencies of such securities in analogous fashion. By canceling mutually offsetting forex order flows, institutions can reduce the required inventory in foreign currencies and thus reduce their risk exposure.

Public market/Alternative Trading System -- A public market with a fully visible order book should operate in a simple FIFO fashion. Optimization algorithms should be used for closed or semi-closed books.

General e-commerce -- Non-financial markets can support a similar mode of operation to contemporary financial markets and thus will be able to benefit from a flexible optimizing crossing system. A prerequisite of such markets (if not used in an auction mode) is a requirement that units being traded are commodities interchangeable with each other and that there is enough liquidity to support real-time operation. Good candidate markets to advance to a real-time exchange, are the various branded micropayment services on the Internet, for example there are a need for an exchange for banner ad imprints or download of copyrighted digital content.

While the invention has been particularly shown and described with reference to preferred embodiments thereof it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of the invention.

CLAIMS

1. An e-commerce system for electronic transactions involving participants having interests in instruments comprising:

storing variables for use in mapping the interests of the participants,

storing constraints for limiting the mapping of interests of the participants,

storing an expression relating the variables,

determining values of the variables that optimize said expression, while satisfying the constraints, thereby mapping the interests of the participants.

2. The e-commerce system of Claim 1 wherein said determining step includes the step of forming an estimate of said values that optimize said expression.

3. The e-commerce system of Claim 2 wherein said determining step continues at least until said estimate is reached.

4. The e-commerce system of Claim 2 wherein said determining step continues until a percentage of said estimate is reached.

5. The e-commerce system of Claim 1 wherein said determining step includes the step of storing values of the variables for one iteration of determining values and using said stored values in a subsequent iteration.

6. The e-commerce system of Claim 1 wherein said determining step includes an optimizing algorithm that calculates toward an optimal result.

7. The e-commerce system of Claim 6 wherein said determining step includes a timing constraint that terminates calculations after a predetermine time.

8. The e-commerce system of Claim 1 wherein said determining step is controlled as a function of the number of transactions.

9. The e-commerce system of Claim 1 wherein said transactions include a number buy orders and sell orders for instruments and said determining step is controlled as a function of the number of buy orders and sell orders.

10. The e-commerce system of Claim 1 wherein said variables include volume and said expression is processed to maximize said volume.

11. The e-commerce system of Claim 1 wherein said variables include spread and said expression is processed to maximize said spread.

12. The e-commerce system of Claim 1 wherein said variables include a total of transactions and said expression is processed to maximize said total.

13. The e-commerce system of Claim 1 wherein said variables include an initiator total of transactions from a distinct transaction initiator and said expression is processed to maximize said initiator total.

14. The e-commerce system of Claim 1 wherein said variables include a fairness threshold for actual orders in transactions.

15. The e-commerce system of Claim 1 wherein said variables include one or more market maker input to provide market maker orders that are available to be used when actual orders are not crossed in transactions.

16. The e-commerce system of Claim 1 wherein said variables include one or more market maker input to provide market maker orders that are available to be used together with actual orders in transactions.

17. The e-commerce system of Claim 1 wherein said variables include one or more market maker inputs to provide market maker orders that are available to improve the price of crossings in combination actual orders in transactions.

18. The e-commerce system of Claim 1 wherein said variables include one or more market maker inputs to provide market maker orders that are available to improve the a number of crossings in combination actual orders in transactions.

19. The e-commerce system of Claim 1 wherein said variables include one or more market maker inputs to provide market maker orders that are available to be used when actual orders are not crossed in transactions.

20. The e-commerce system of Claim 1 wherein said constraints include a fairness input to specify bounds on executions.

21. The e-commerce system of Claim 20 wherein said fairness input is a price quote from at least one external market and orders are crossed only when the crossing exceeds the price quote.

22. The e-commerce system of Claim 1 wherein said transactions are executed by a market engine that includes a number of internal orders for instruments and that includes a number of external orders for instruments.

23. The e-commerce system of Claim 22 wherein said transactions are optimized on said internal orders.

24. The e-commerce system of Claim 22 wherein said transactions are optimized on said external orders.

25. The e-commerce system of Claim 22 wherein said transactions are optimized on both said internal orders and said external orders.

26. The e-commerce system of Claim 1 further storing two or more algorithms for executing said determining values step and wherein said determining values step selects one of said algorithms.

27. The e-commerce system of Claim 1 wherein said constraints include a trigger input for triggering a crossing for a transaction.

28. The e-commerce system of Claim 27 wherein said trigger input is a function of a number of orders pending.

29. The e-commerce system of Claim 27 wherein said trigger input is a function of a volume of orders processed.

30. The e-commerce system of Claim 27 wherein said trigger input is a function of time.

31. The e-commerce system of Claim 1 wherein said constraints include a trigger input for evicting orders.

32. The e-commerce system of Claim 31 wherein said trigger input is a function of time.

33. The e-commerce system of Claim 1 further storing two or more algorithms for executing said determining values step and wherein said determining values step selects one of said algorithms for each symbol.

34. The e-commerce system of Claim 1 wherein orders are tracked by instrument symbols and said system further storing two or more algorithms for executing said determining values step and wherein said determining values step selects one of said algorithms for each symbol.

35. The e-commerce system of Claim 34 wherein orders are of different order types including market orders and limit orders and said system further stores two or more algorithms for executing said determining values step and wherein said determining values step selects one of said algorithms as a function of order type.

36. The e-commerce system of Claim 34 wherein orders are of different arithmetic types including fractions and said system further executes said determining values step independent of said arithmetic type.

37. An e-commerce system for electronic transactions involving participants having interests in instruments comprising:

one or more transaction units,

network means connected to the transaction units,

one or more market engines connected to said network means, each of said market engines and including,

a plurality of components for processing transactions where said components include at least:

one or more transaction unit interface components for interfacing with transaction units;

one or more execution components for executing transactions;

a crossing component including,

means for storing variables for use in mapping the interests of the participants,

means for storing constraints for limiting the mapping of interests of the participants,

means for storing an expression relating the variables,

means for determining values of the variables that optimize said expression, while satisfying the constraints, thereby mapping the interests of the participants.

a connection element connecting said components.

38. The system of Claim 37 wherein each of said plurality of components includes,

one or more computers,

one or more operating systems executing on said one or more computers, respectively,

application processes executing on said one or more computers under control of said one or more operating systems, said application processes including,

a function application for executing functions,

a communication application for controlling communication among said components,

a resource management component for controlling the allocation of processes among said one or more computers.

39. The system of Claim 39 wherein said plurality of execution components includes one or more of the following components:

a routing component, a trigger component, a crossing component, a scripting component, a stock component, a bond component, a currency component, an options component, an accounting component, a storage component, and a supervisor component.

40. The system of Claim 39 wherein said interface components include one or more of the following components:

a transaction interface component, a transaction processing interface component and a data access interface component.

41. The system of Claim 37 wherein said components include a routing component for routing transactions in said e-commerce system.

42. The system of Claim 37 wherein said components include,

a storage component for providing storage in said e-commerce system of internal transactions,

a crossing component for internal crossing of said internal transactions,

a routing component, for routing transactions in said e-commerce system, operating to determine available external transactions and enabling said crossing component to operate to cross said internal transactions when the internal transactions are fair with respect to said external transactions.

43. The system of Claim 37 wherein said components include a trigger component for processing conditions in connection with said transactions.

44. The system of Claim 37 wherein said components include a crossing component for crossing transactions internal to said market engine.

45. The system of Claim 37 wherein said components include a scripting component for scripting transactions in said e-commerce system.

46. The system of Claim 37 wherein said components include a stock component for shopping for stock transactions in said e-commerce system.

47. The system of Claim 37 wherein said components include a bond component for shopping for bond transactions in said e-commerce system.

48. The system of Claim 37 wherein said components include a currency component for currency exchanging in said e-commerce system.

49. The system of Claim 37 wherein said components include an option component for shopping options in said e-commerce system.

50. The system of Claim 37 wherein said components include an accounting component for providing accounting functions in said e-commerce system.

51. The system of Claim 37 wherein said components include a storage component for providing storage in said e-commerce system.

52. The system of Claim 37 wherein said components include a supervisor component for supervising operations of the market engine in said e-commerce system.

53. The system of Claim 37 wherein said market engines include a plurality of groups of market engines wherein each market engine in one of said groups includes one or more of said components for controlling operation in said group.

54. The system of Claim 53 wherein said one or more of said components for controlling operation in said group is a routing component for routing transactions in said e-commerce system in accordance with operations common to said group.

55. The system of Claim 37 wherein said e-commerce system includes a plurality of local offices, each local office having a plurality of workstations and a concentrator for connection through said network means to a legacy office and to said one or more market engines.

56. The system of Claim 37 wherein said transaction units include,
one or more external transaction initiators for initiating a transaction,
a plurality of external transaction processors for processing transactions.

57. The system of Claim 37 wherein,
each of said plurality of components includes,
one or more computers, one or more operating systems executing on said one or more computers, respectively, and application processes executing on said one or more computers under control of said one or more operating systems, said application processes including, a function application for executing a function, a communication application for controlling communication among said components, and a resource management component for controlling the allocation of processes among said one or more computers,
and wherein,
a first one of said plurality of components is a transaction initiator component having said function application as a transaction initiator application for providing instruments for crossing,
a second one of said plurality of components is a data access component having said function application as a data access application for providing external data from external data sources,
a third one of said plurality of components is a storage component having said function application as a order book application for storing instruments received from said transaction initiator component under control of said communication application,

a fourth one of said plurality of components is a crossing component having said function application as a crossing application and for receiving said instruments for crossing from said storage component and said external data from said data access component under control of said communication application, said crossing component operating to cross said instruments under conditions based upon said external data.

58. The system of Claim 57 wherein,

a fifth one of said plurality of components is a transaction processor interface component for interfacing transaction to external transaction processors,

a six one of said plurality of components is a routing component for routing transactions to external processors through said transaction processor interface component under control of said communication application when transactions have not been crossed by said crossing component.

59. The system of Claim 37 wherein,

each of said plurality of components includes,

one or more computers, one or more operating systems executing on said one or more computers, respectively, and application processes executing on said one or more computers under control of said one or more operating systems, said application processes including, a function application for executing a function, a communication application for controlling communication among said components, and a resource management component for controlling the allocation of processes among said one or more computers,

and wherein,

a first one of said plurality of components is a transaction initiator component having said function application as a transaction initiator application for providing instruments for crossing,

a second one of said plurality of components is a storage component having said function application as a order book application for storing instruments received from said transaction initiator component as internal instruments under control of said communication application,

a third one of said plurality of components is a crossing component having said function application as a crossing application and for receiving said internal instruments for crossing from said storage component under control of said communication application, said crossing component operating to cross said internal instruments when enabled.

60. The system of Claim 59 wherein,

a fourth one of said plurality of components is a transaction processor interface component for interfacing transaction to external transaction processors,

a fifth one of said plurality of components is a routing component for routing transactions to external processors through said transaction processor interface component under control of said communication application, said routing component operating to determine available external transactions and enabling said crossing component to operate to cross said internal transactions when the internal transactions are fair with respect to said external transactions.

61. In an e-commerce system formed of a plurality of components where each component includes one or more computers, one or more operating systems executing on said one or more computers, respectively, and application processes executing on said one or more computers under control of said one or more operating systems, said application processes including, a function application for executing a function, a communication application for controlling communication among said components, and a resource management component for controlling the allocation of processes among said one or more computers, the steps comprising,

providing instruments for crossing,

storing instruments as internal instruments,

detecting external instruments available for crossing,

internally crossing said internal instruments if internal crossing is as fair as crossing said internal instruments with external instruments,

externally crossing said internal instruments with external instruments if the internal instruments are not internally crossed.

FIG. 1

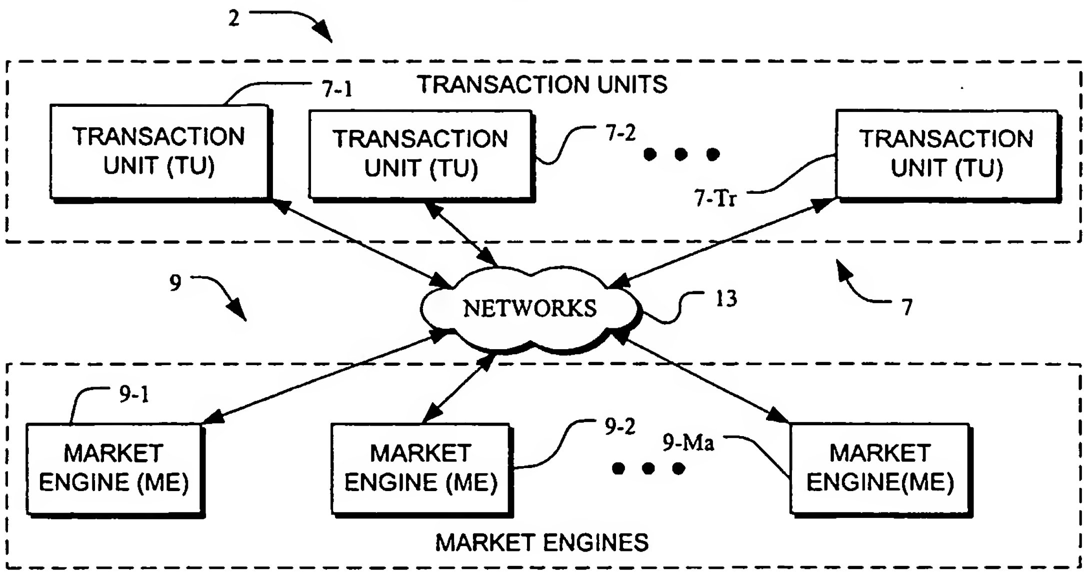


FIG. 2

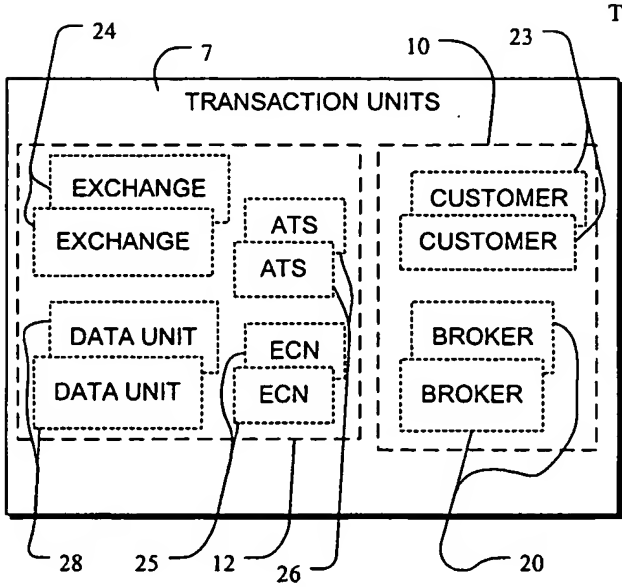


FIG. 3

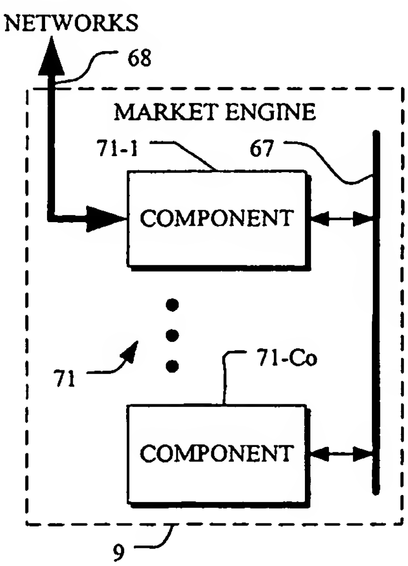


FIG. 4

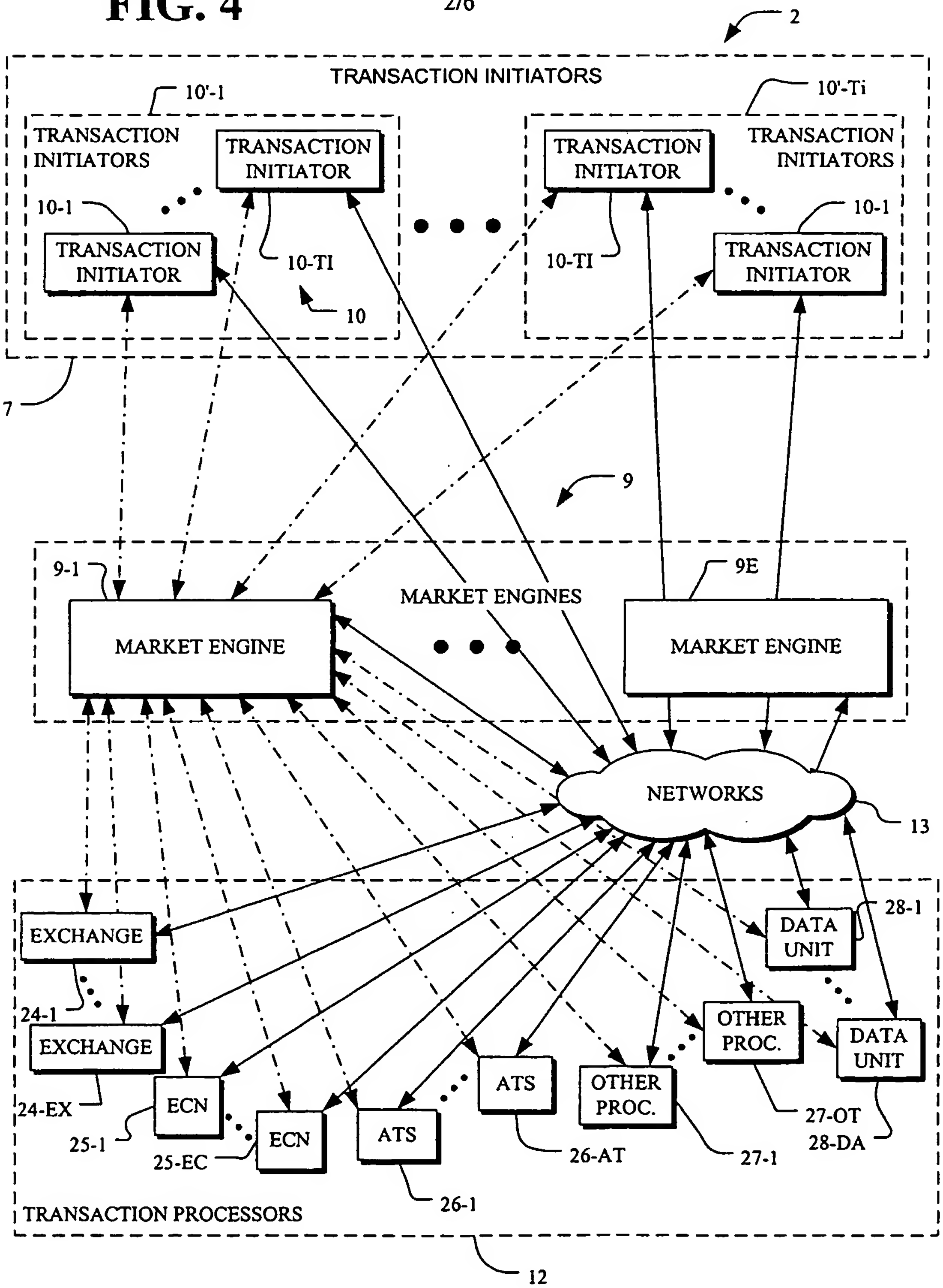
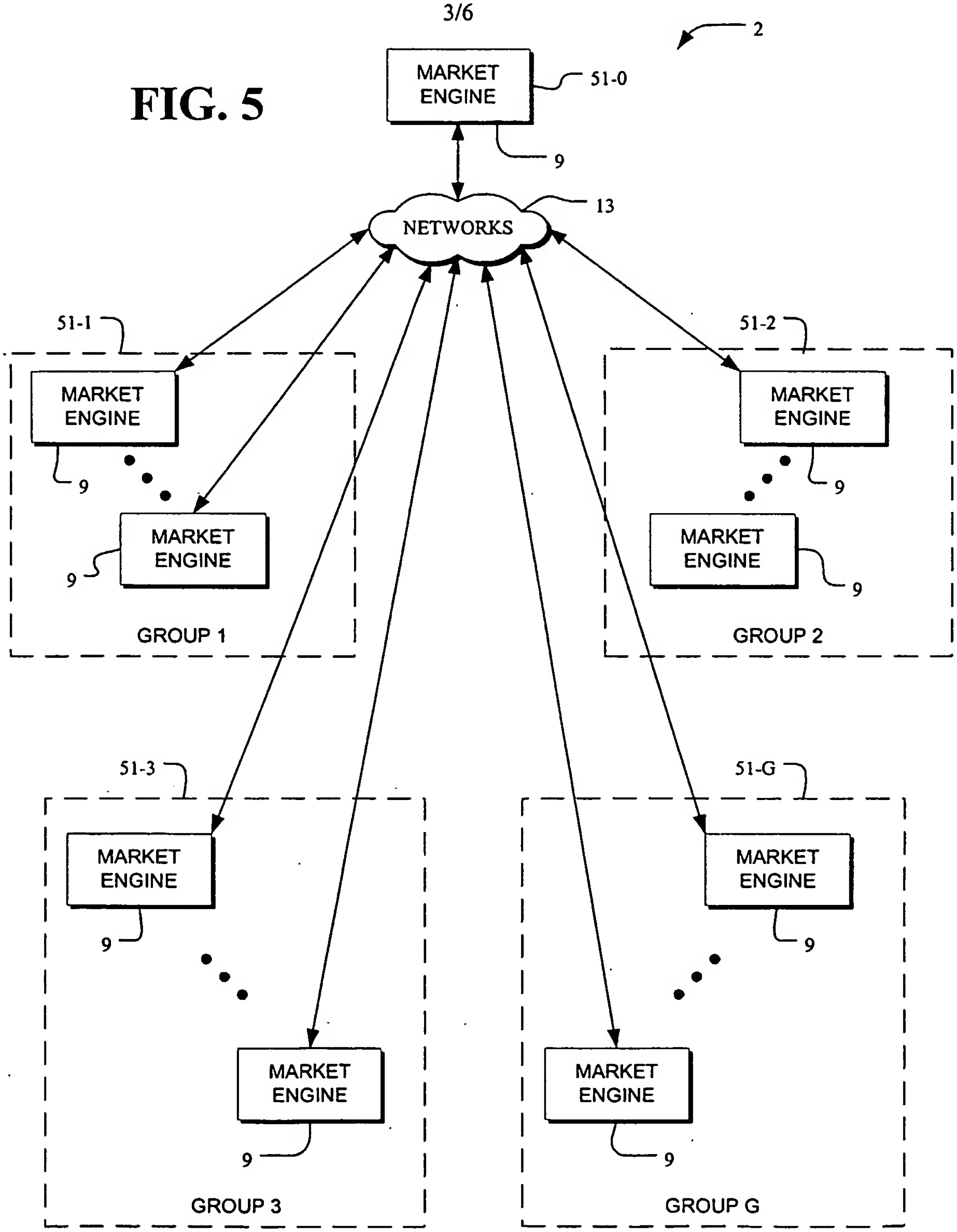

2
TRANSACTION INITIATORS
10'-1
10'-Ti
TRANSACTION INITIATORS
TRANSACTION INITIATOR
TRANSACTION INITIATOR
TRANSACTION INITIATORS
10-1
TRANSACTION INITIATOR
10-TI
10
10-TI
10-1
TRANSACTION INITIATOR
7
9
9-1
9E
MARKET ENGINES
MARKET ENGINE
MARKET ENGINE
NETWORKS
13
28-1
EXCHANGE
DATA UNIT
24-1
EXCHANGE
OTHER PROC.
DATA UNIT
ATS
OTHER PROC.
24-EX
ECN
25-1
ECN
ATS
27-OT
28-DA
25-EC
26-AT
27-1
26-1
TRANSACTION PROCESSORS
12

FIG. 5


2
MARKET ENGINE
51-0
9
NETWORKS
13
51-1
MARKET ENGINE
9
MARKET ENGINE
9
GROUP 1
51-2
MARKET ENGINE
9
MARKET ENGINE
9
GROUP 2
51-3
MARKET ENGINE
9
MARKET ENGINE
9
GROUP 3
51-G
MARKET ENGINE
9
MARKET ENGINE
9
GROUP G

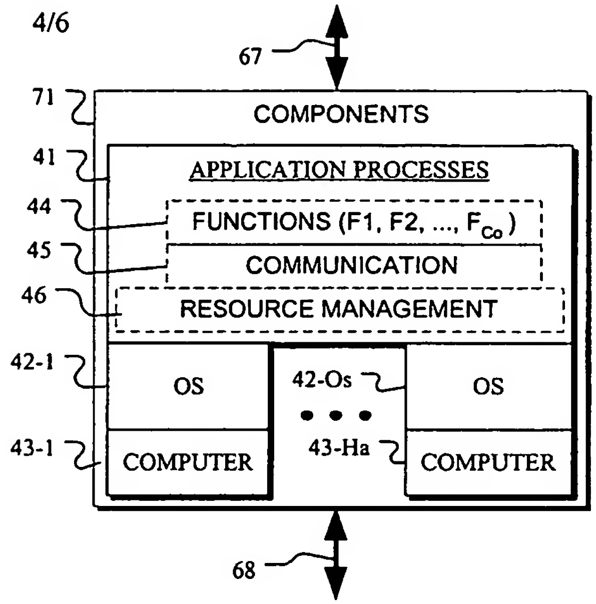

67
71
41
44
45
46
42-1
43-1
COMPONENTS
APPLICATION PROCESSES
FUNCTIONS (F1, F2, ..., F_{Co})
COMMUNICATION
RESOURCE MANAGEMENT
OS
42-Os
OS
COMPUTER
43-Ha
COMPUTER
68

FIG. 6

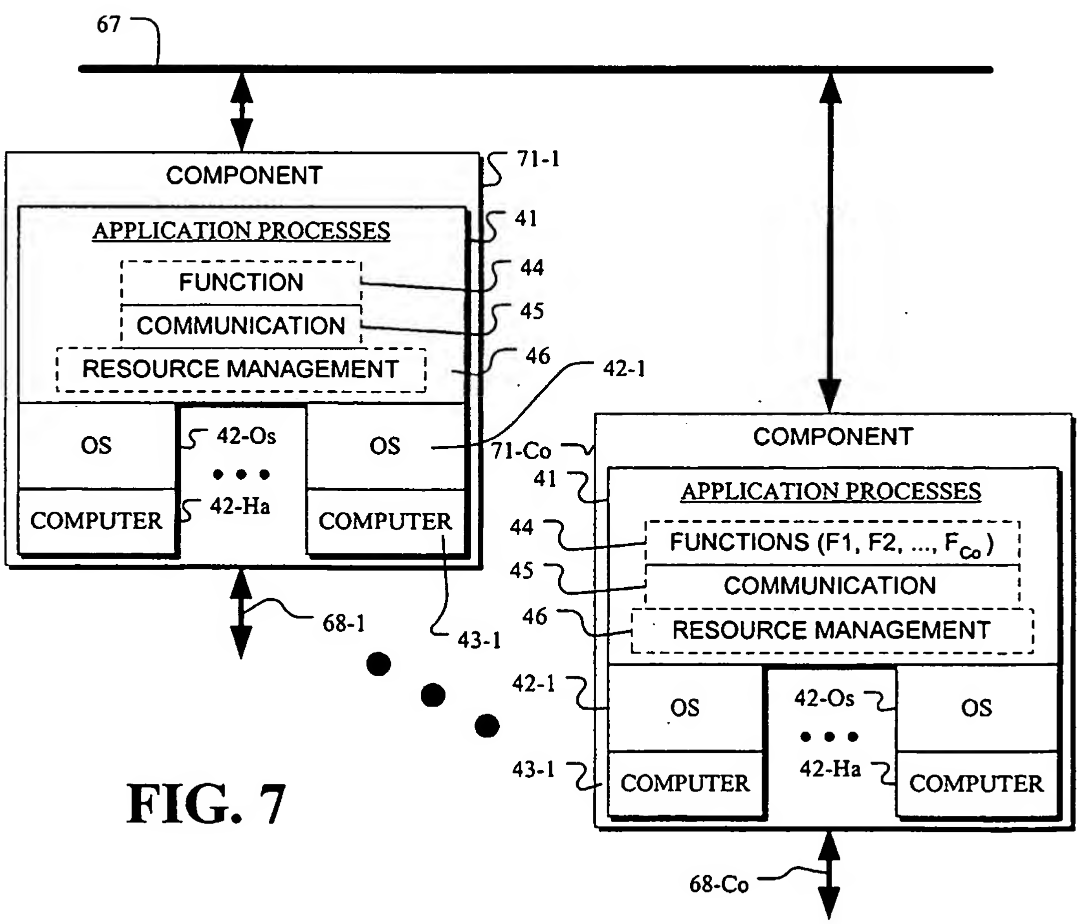

67
COMPONENT
71-1
APPLICATION PROCESSES
41
FUNCTION
44
COMMUNICATION
45
RESOURCE MANAGEMENT
46
42-1
OS
42-Os
OS
COMPUTER
42-Ha
COMPUTER
68-1
43-1
71-Co
COMPONENT
41
APPLICATION PROCESSES
44
FUNCTIONS (F1, F2, ..., F_{Co})
45
COMMUNICATION
46
RESOURCE MANAGEMENT
42-1
OS
42-Os
OS
43-1
COMPUTER
42-Ha
COMPUTER
68-Co

FIG. 7

FIG. 8

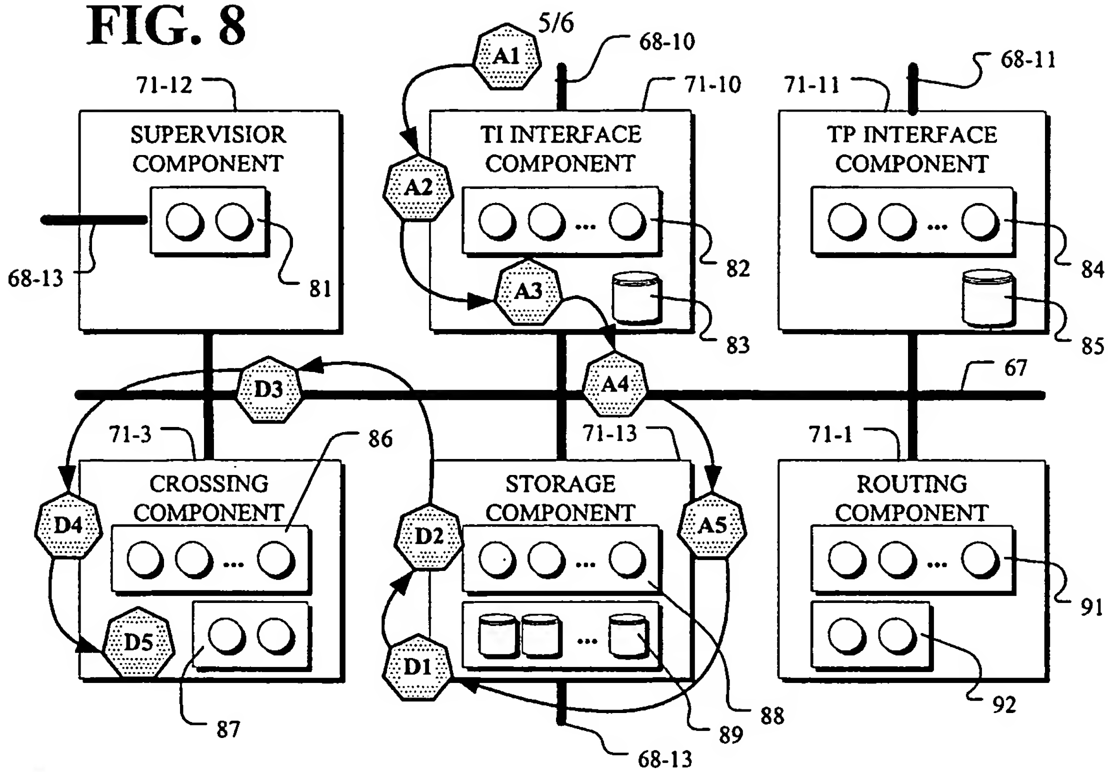

SUPERVISIOR COMPONENT
TI INTERFACE COMPONENT
TP INTERFACE COMPONENT
CROSSING COMPONENT
STORAGE COMPONENT
ROUTING COMPONENT
A1
A2
A3
A4
A5
D1
D2
D3
D4
D5
68-10
68-11
68-13
71-12
71-10
71-11
71-3
71-13
71-1
67
81
82
83
84
85
86
87
88
89
91
92

FIG. 9

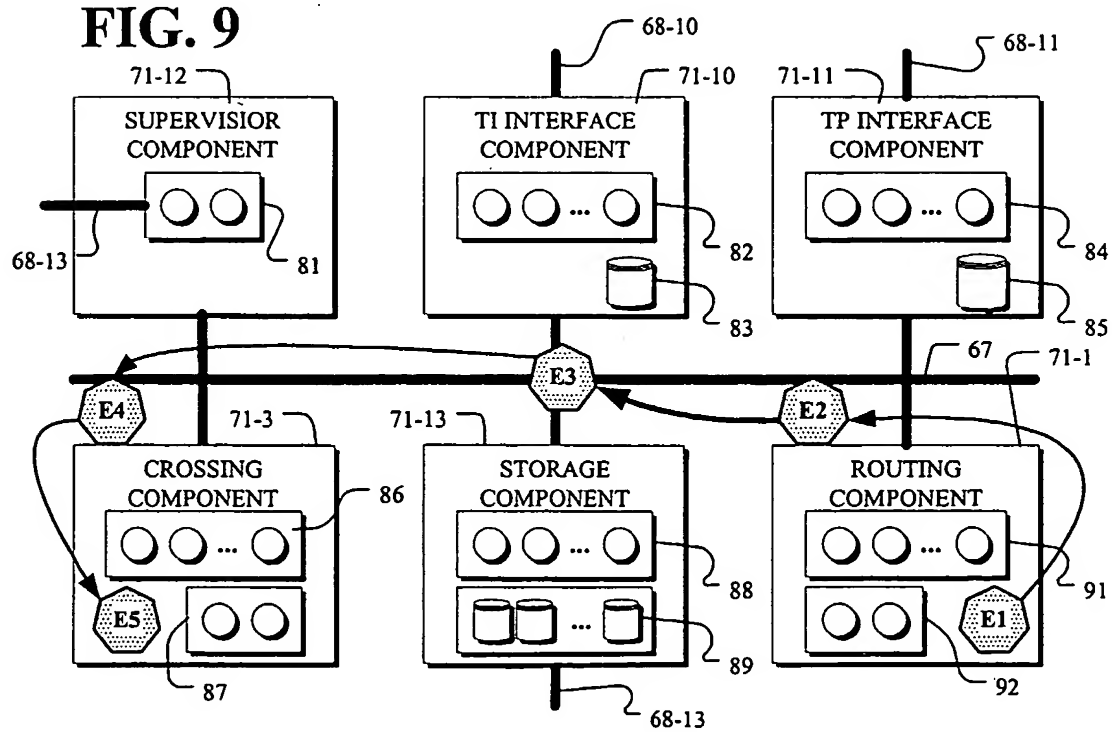

SUPERVISIOR COMPONENT
TI INTERFACE COMPONENT
TP INTERFACE COMPONENT
CROSSING COMPONENT
STORAGE COMPONENT
ROUTING COMPONENT
E1
E2
E3
E4
E5
68-10
68-11
68-13
71-12
71-10
71-11
71-3
71-13
71-1
67
81
82
83
84
85
86
87
88
89
91
92

FIG. 10